64 PLUS 4



& AMIGA

CZERWIEC 1991 ISSN 0867-3918 INDEKS 377112 CENA 5000 zł

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



WITAJCIE WAKACJE!

W numerze :

W następnym numerze :

- BLITTER
- TRACK WRITE V1.02
- POLICE Quest
- Cheats C-64

OGŁOSZENIA



miesięcznik nr 6(8) czerwiec 1991
cena 1 egz.: 5000 zł



Wydawca: ABUK Spółka z o. o.
Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.

Adres redakcji: Redakcja „64 plus 4" 85-166 Bydgoszcz 43 skrytka pocztowa 64
Redagują: Marcin Dudar, Sambor Kuźma, Paweł Sołtysiński, Waldemar Szczygieł (red. nacz.), Robert Turliński, Wojciech Wasilewski.
Okładka: Sławomir Karolczak.
Skład: ABUK
Druk: Z.P. POLRASTER 85-353 Bydgoszcz, ul. Orawska 19.

Drodzy Czytelnicy!

Nasze pismo od początku powstawało i powstaje w gronie miłośników komputerów Commodore. Zaczynając je wydawać nie posiadaliśmy ani wielkiego zaplecza finansowego, ani wieloletnich doświadczeń w pracy redakcyjnej.

Zainteresowanie naszym pismem przerosło nasze oczekiwania na tyle, że szybko musieliśmu się zreorganizować. Mimo naszych starań niektórzy z Was bardzo długo oczekiwali na realizację swoich zamówień, posądzając nas nawet o nieuczciwość. Jeszcze raz chcielibyśmy wszystkich przeprosić za wszelkie opóźnienia. Aktualnie wszystkie zamówienia realizowane są natychmiast po otrzymaniu z banku informacji o wpłacie.

Przy okazji prosimy bardzo o czytelne podawanie swojego imienia, nazwiska, adresu oraz przedmiotu zamówienia na blankietach. Zdarza się bowiem dosyć często, że otrzymujemy dowód wpłaty tylko z podpisem lub bez określenia czego wpłata dotyczy. W takich przypadkach zmuszeni jesteśmy do oczekiwania na ponowny, listowny kontakt.

W instrukcji Voicetracker'a znajduje się informacja, że do programu dołączonych jest 35 demonstracji muzycznych, tymczasem w wersji taśmowej jest ich znacznie mniej - dlaczego? - pyta kilku czytelników. Otóż instrukcja została opracowana dla wersji dyskowej.

Początkowo nie planowaliśmy sprzedaży VV 4.0 na taśmie. Dopiero Wasze listy wskazały nam potrzebę opracowania takiej wersji. Długość kaset nie pozwalała na umieszczenie wszystkich demonstracji. Stąd ta niezgodność z informacją zawartą w instrukcji. Należy zaznaczyć, iż w kalkulacji ceny zarówno w przypadku wersji dyskowej, jak i taśmowej wszystkie demonstracje uznawane były jako BEZPŁATNE.

W związku z licznymi sugestiami dotyczącymi wydawania taśmowych wersji programów Public Domain informujemy, że przygotowujemy kasetę z wybranymi programami, jakie do tej pory ukazały się w dyskowej wersji PDP. O szczegółach - w następnym numerze!

Do zobaczenia za miesiąc!

Redakcja

Z daleka i z bliska

- Duddie zakończył pracę nad D-Mon'em v 1.8 ! Z dobrze poinformowanych źródeł wiemy, że istnieje poważna szansa na wydanie D-Mon'a przez firmę Data Becker.

- Redaktorzy pisma „64 plus 4" otrzymali propozycję wydawania pisma dyskowego „KEBAB" jako magazynu komercjalnego. Czy wszystko się powiedzie ? Czas pokaże.

- W dniach 25 - 28 kwietnia w Berlinie odbyła się wielka wystawa Amiga Exhibition ! Szukajcie w numerze relacji.

- Z radością informujemy, że w tym miesiącu nie udało nam się złapać żadnego nowego wirusa !

- Powstał nowy magazun dyskowy na C-64 o nazwie „Hi-Five". Jest wydawany przez grupę Mystic w kooperacji z The Force. Pierwsze wydanie zawiera wiele nowin z australijskiej sceny, opisy dem i gier. Autorzy zaproponowali konkurs na najlepszą muzyczkę. Wyniki będą ogłoszone 1 czerwca. Postaramy się sprowadzić najlepsze z najlepszych i zamieścić je na naszym PDP.

- Powstał nowy assembler. Nazywa się „Easy Expert" i jest przeznaczony dla Commodore 64 z rozszerzeniem pamięci do 256 KB. Pamięć jest podzielona na cztery banki po 64 KB. Znajdują się w nich kolejno: sam assembler, kod źródłowy, dane do grafiki i kod wynikowy.

- Na jednym z dysków znaleźliśmy program autorstwa Nico Francois o nazwie Disk Talk. Umożliwia on podłożenie i odtworzenie dowolnych sampli przy wkładaniu i wyjmowaniu dyskietki ze stacji. Chyba ludzie bardzo się nudzą...

- Rozpadła się grupa WFMH (World Federation of Mad Hackers). Thorgal, Nugie, Thunder, Dr. Cloo, Cadaver przeszli do grupy AMAZE.

- Nasi wchodzą na światową scenę. O Katharsis już głośno a na pewno nowe grupy z Trójmiasta również dadzą o sobie znać (The Luzers Team i Shadow in Tricity). Na razie przeczytaliśmy ciekawy artykuł Mr.Root'a (Katharsis) o polskiej scenie CB-Radiowców (ZINE #9).

- Ludzie, cieszcie się, że nie macie modemów! Lider grupy Angels został zmuszony do zaprzestania działaności przez... rachunek telefoniczny opiewający na 3500 dolarów !

- Uwaga ! Jeśli mieliście kontakt z adresem PLK 145324 E W-4787 Geske 1 w Niemczech to już nie piszcie ! Adres jest spalony i policja może też was przyskrzynić.

- Z dziwiątego numeru ZINE dowiedzieliśmy się o nowym wirusie o wdzięcznej nazwie (ha!) „SADDAM". Może ktoś nam dobrowolnie podrzuci ?!!

- W sierpniu w Norwegi odbędzie się kolejny Amiga-Show.
  Czas: 8-11 sierpień 1991r.
  Nazwa: Amiga Camp'91.
  Organizator: Tessa Elektronics.
  Ceny: Ponoć niskie.
  Zabawa: Jak zwykle świetna (oby).

- Sławna i legendarna grupa KEFRENS rozpadła się definitywnie.

- Orion koduje obecnie ostatnie demo. Szkoda, będzie nam brakowało waszych programów.

- Ostatnimi dniami ukazał się na rynku nowy program firmy Intelligent Memory o nazwie Imagine. Program umożliwia tworzenie animacji 3-D z zastosowaniem ray-traceing'u. Jego możliwości są oszałamiające i wprost proporcjonalne do jego ceny, która (bagatela!) wynosi 598 DM.

opr. Hi-Man

AMI-Show - Berlin 91

Ciekawy jestem ilu z posiadaczy Amigi w Polsce wiedziało o odbywającej się w Berlinie wystawie poświęconej wyłącznie komputerom tego typu.

Wystawa odbyła się w dniach 26-28 kwietnia tego roku w berlińskiej hali Messegelande AMK i skupiła dziesiątki producentów sprzętu, oprogrmowania i literatury dotyczącej wszystkich typów Amig.

Informację o planowanej wystawie otrzymałem od członków mojej grupy (Padua) mieszkających w Berlinie.

W sobotę, 28 kwietnia, po niewielkich perypetiach na granicy (alkohol? papierosy? nic?!) dotarłem wraz z przyjaciółmi do Centrum Kongresowego w Berlinie.

Bliet wstępu, ważny przez całą wystawę, kosztował 17DM. Cóż ... kupiliśmy, weszliśmy.

Rozmieszczenie stanowisk i ich wygląd nie odbiegał chyba za bardzo od naszych wyobrażeń dotyczących tego typu imprez (jeśli ktoś był na Targach Poznańskich, to wie jak to wygląda).

A jakich firm tam nie było! Z bardziej znanych można wymienić Ubi-Soft, Supra, Delta Concept, Vortex, KingSoft, Precision Software, Mukra Datentechnik, Software 2000, KCS czy Leisure Soft.

Obecnych było również kilka wydawnictw poświęconych Amidze: Amiga Magazin, Amiga Dos, Markt and Technik Verlag. Był nawet specjalny samochód - symulator na hydraulicznych wysięgnikach, we wnętrzu którego można było sobie pograć w OUT RUN, był (niestety kiepski) laser-show, reklamujący jednego ze sponsorów wystawy - berlińską firmę FAB. Na stoisku Amiga Dos Chris Huelsbeck podpisywał autorskie kopie programu muzycznego TFMX na Amigę.

Na kilku stoiskach z drobnym osprzętem można było kupić okazyjnie dyskietki (noname 5.25" - 4DM; noname 3.5" - 7DM), myszki (40 DM) lub rozszerzenie pamięci A500 do 1Mb z podtrzymywanym akumulatorowo zegarem (za jedyne 69 DM).

W samym środku sali znajdowało się pokaźnych rozmiarów centrum informacyjne Commodore, które organizatorzy otoczyli ze wszystkich stron wysokim (ponad dwa metry) parkanem, na którym można było uwiecznić fakt swojego przyjazdu: „Tu byłem! Józek". Cały ów parkan pokryty był przez nazwy grup, które się tu znajdowały. Materiały i narzędzia: najczęściej fluoroscencyjne markery, grafitti lub nawet nalepki o formatach od A6 do A1!

Najciekawsze grupy, które były w sobotę: Red Sector, Phenomena, Death Sector, Targets, The Wanted, Lower Level, Giants i wielu innych. A wszyscy chodzili opieczętowani. O co chodzi?

Chodzi o pewien pomysł organizatorów (nie wiem czy nowy, czy stary...), który polegał na uporządkowaniu problemu tych osób, które musiały na moment wyjść z terenu wystawy, a potem wrócić. Wielu zwiedzających (m. in. i my) z tego korzystało choćby dlatego, że jedzenie w restauracji mieszczącej się na terenie wystawy było bardzo drogie (jak stwierdzali ze zgrozą Niemcy - prawie trzykrotnie droższe niż w zwykłych jadłodajniach!).

Otóż organizatorzy, aby uniknąć przekazywania sobie wzajemnie biletów, wprowadzili okolicznościowe pieczątki, które stawiano na dłoni każdemu wychodzącemu z wystawy. Powracając można już było wejść bez żadnych problemów.

Na wystawie prezentowano ponownie najmłodsze dziecko Commodore - Amigę CDTV. Obserwujący barwny pokaz albo cmokali pod nosem, albo stukali się w czoło. Komentarze były mniej więcej w tym stylu:„kombinują, kombinują, a potem znowu się to na rynku nie przyjmie i będą straty".

Na stoiskach różnych firm prezentowano min. karty o wspólnej nazwie „flicker free video", które zainstalowane w A500, A1000 lub A2000 usuwają przykry efekt migotania w trybie „interlace". Najtańszą taką przystawkę oferowała firma ICD za 899DM.

Firma California Access przedstawiła swój kontroler do dysku twardego 3,5' (do A2000), który, jak reklamowano, oferuje 16 bitowy transwer danych, w praktyce oznacza to transmisję 700 Kb na sekundę.

A teraz coś dla miłośników grafiki: firma M.A.S.T. zaprezentowała kartę do wszystkich Amig z minimum 1MB pamięci RAM o nazwie Colorburst. Karta ta posiadając własny procesor graficzny (Custom VLSI) i 1.5MB własnej RAM oferuje 16.8 miliona kolorów (!), 48 bitplanes, opis każdego punktu na ekranie za pomocą 24 bitów, 24 wirtualne sprite'y. Obrazy przez nią generowane mogą być wyświetlane „z przodu" lub „z tyłu" orginalnego obrazu Amigi, albo jednocześnie z obu stron. Rozdzielczość od 384x580 do 768x580 (480 w NTSC). Twórcy oświadczali, że dzięki specjalizowanemu procesorowi graficznemu animacja limitowana jest tylko przez szybkość samej Amigi i jej peryferiów.

O podobnych możliwościach kartę (brak procesora, większa rozdzielczość) zaprezentowała firma ACS Harlequin.

Na wystawie było wiele okazji do poznania nowych, ciekawych ludzi, do wymiany poglądów i doświadczeń.

Paweł Sołtysiński

MAPA PAMIĘCI – CZĘŚĆ VI

Artykuł ten kończy opis szczegółowej mapy pamięci komputerów rodziny C16 / C116 / Plus/4. Przedstawiamy w nim rejestry specjalizowanego układu we/wy TED, a za miesiąc ostatni fragment tabeli opisującej funkcje poszczególnych komórek pamięci.

Najciekawszym spośród zastosowanych w komputerach rodziny C16 układów jest specjalizowany chip we/wy - TED. Układ ten jest odpowiedzialny za wyświetlanie obrazu TV, generowanie dźwięku, obsługuje joystick i klawiaturę. Jego pracą steruje się poprzez wpisywanie odpowiednich wartości do 32 rejestrów.

W typowy dla komputerów opartych na procesorach z rodziny 65XX (i pochodnych) sposób rejestry układów we/wy są włączone w przestrzeń adresową pamięci procesora. Dostęp do nich odbywa się identycznie, jak do komórek pamięci - w języku BASIC za pomocą instrukcji PEEK i POKE. Ponieważ, jak widać w załączonej tabeli, poszczególnymi funkcjami sterują wartości pojedyńczych bitów rejestru, potrzebny jest sposób ustalania stanu poszczególnych bitów bez zmiany pozostałych. W języku BASIC najwygodniej jest posługiwać się instrukcjani PEEK i POKE korzystając z pomocy funkcji AND i OR.

Zanim omówię to dokładniej, przedstawię pojęcie „wagi bitu". Waga bitu jest to wartość, jaką reprezentuje dany bit w bajcie. Wagi poszczególnych bitów wynoszą :

bit 0 = 1   bit 1 = 2   bit 2 = 4   bit 3 = 8
bit 4 = 16   bit 5 = 32   bit 6 = 64   bit 7 = 128

Suma wag wszystkich bitów wynosi 1+2+4+8+16+32+64+128 = 255

W ten sposób można obliczyć wartość bajtu sumując wagi poszczególnych bitów, które są ustawione (mają wartość 1) oraz podstawiając wartość 0 dla bitów, które są wyzerowane. Najlepiej wyjaśni to przykład:

```
bit:        7 6 5 4 3 2 1 0
stan bitu:  0 0 1 0 1 1 0 1
wartosc:    0+0+32+0+8+4+0+1=45
```

Aby ustawić pojedynczy bit w rejestrze o adresie X należy wykonać sekwencję rozkazów:

```
10 POKE X, PEEK(X) OR waga
```

Chcąc właczyć tryb „extended color" (patrz artykuł Jarosława Rafy „Grzebanie w bajtach" w poprzednim numerze 64 PLUS 4) należy ustawić bit 6 rejestru 6 TED (adres 65286). Waga tego bitu wynosi 64. Piszemy więc:

```
20 POKE 65286, PEEK (65286) OR 64
```

Aby wyzerować dany bit w rejestrze o adresie X należy wykonać sekwencję rozkazów:

```
30 POKE X, PEEK (X) AND (255-waga)
```

Chcąc przywrócić normalny tryb (wyłączyć „extended color") należy wyzerować bit 6 rejestru 6:

```
40 POKE 65286, PEEK (65286) AND 191
 : REM 255-64=191
```

Aby ustalić jednocześnie stan kilku bitów należy w miejsce wartości waga podstawic sumę wag bitów, których stany mają być ustalone. Aby włączyć na stałe generatory 1 i 2 (bity 4 i 5 rejestru 17), należy po ustaleniu ich częstotliwości instrukcją SOUND oraz głośności instrukcją VOL wpisać:

```
50 POKE 65297, PEEK (65297) OR 48 : REM 16+32=48
```

Podobnie, chcąc ustawić na zero głośność, należy wyzerować bity 0-3 rejestru 17. Aby to zrobić piszemy:

```
60 POKE 65297, PEEK (65297) AND 240
 :REM 255-(1+2+4+8)=240
```

Poza użyciem instrukcji SOUND i VOL częstotliwość i głośność dźwięku można ustalić również zmieniając zawartość rejestrów:

częstotliwość: rejestr 14, 15, 16 (bit 0-1) oraz 18 (bit 0-1).
głośność: rejestr 17 (bit 0-3)

Tym sposobem można zmieniać częstotliwość dźwięku w czasie jego trwania:

```
70 SOUND 1,1000,3000
80 FOR J=0 TO 255
90 POKE 65294,J:NEXT
100 SOUND 1,1,0: REM wyłącza generator
```

I na koniec pewna ciekawostka. Otóż mój Plus/4 pierwotnie pracował w systemie telewizyjnym NTSC. W systemie tym obraz ma 525 linii, a częstotliwość obrazu wynosi 60 Hz. W obowiązującym u nas systemie (a także w systemie PAL) obraz ma 625 linii, częstotliwość obrazu wynosi 50 Hz. Mimo tych różnic udało się tak wyregulować monitor Neptun 156, że pracował poprawnie zarówno z komputerem Plus/4 jak i Timex (system PAL). W systemie NTSC obraz był węższy w pionie. Po pewnym czasie okazało się, że niektóre gry nie działają poprawnie - dźwięk jest dobry, a obraz nieczytelny. Po przeanalizowaniu kodu gry okazało się, że jej autor założył pracę komputera wyłącznie w systemie PAL. W związku z tym ustalając tryb graficzny program wpisywał nową wartość do rejestru 7 TED - bez sprawdzenia poprzedniej. W tej nowej liczbie bit 6 był wyzerowany, co powodowało przejście układu TED w tryb pracy PAL. Jednak generowany sygnał TV nie był prawdziwym sygnałem PAL, ponieważ w komputerach pracujących w tym systemie oscylator generuje inną częstotliwość. Ponieważ przerabianie wszystkich programów tak, aby pracowały na każdym komputerze jest dosyć niewygodne, więc przerobiłem komputer. Aby komputer pracował w systemie PAL, oprócz rezonatora kwarcowego musiałem rownież wymienić pamięć ROM z systemem operacyjnym. Zawarta w oryginalnej pamięci procedura startu ustawiała układ TED w tryb NTSC, co wymagało ręcznego przełączania trybu przy każdym włączeniu komputera i wpisywania ciągle „POKE 65287, PEEK (65287) AND 191"

Cykl artykułów poświęconych mapie pamięci komputerów rodziny C16 dobiegł końca. Celem moim było ułatwienie wszystkim użytkownikom własnych eksperymentów z programowaniem tych ciekawych maszyn. Mam nadzieję, że cel ten został osiągnięty.

Wojciech Wasilewski

REJ	HEX	DEC	BIT	OPIS
0	FF00	65280		timer #1 LSB
1	FF01	65281		timer #1 MSB
2	FF02	65282		timer #2 LSB
3	FF03	65283		timer #2 MSB
4	FF04	65284		timer #3 LSB
5	FF05	65285		timer #3 MSB
6	FF06	65286	0-2	pionowa pozycja scroll, dla normalnego ekranu 25-wierszowego zawiera wartość 3
			3	wybór 24/25 wierszy (1=25 wierszy)
			4	1=ekran włączony, 0=wyłączony
			5	1=grafika HIRES włączona
			6	1=tryb extended color włączony
			7	bit testowy, musi być 0
7	FF07	65287	0-2	pozioma pozycja scroll, 0=brak przesuwu. Aby przesuw był możliwy bit 3 tego rejestru =0
			3	wybór 38/40 kolumn (1=40 kolumn)
			4	1=tryb wielobarwny włączony
			5	1=tryb freeze włączony
			6	0=PAL 1=NTSC
			7	RVS-video 0=sprzętowe, 1=programowe
8	FF08	65288		rejestr klawiatury
9	FF09	65289		rejestr źródła przerwań
			0	nie używany
			1	raster
			2	pióro świetlne
			3	timer #1, zmniejsza się do zera
			4	timer #2, zmniejsza się do zera
			5	nie używany
			6	timer #3
			7	wskaźnik przerwań
10	FFOA	65290		rejestr maski przerwań
			0	bit 8 z rejestru 11
			1-6	tak samo jak w rejestrze 9
			7	nie używany
11	FFOB	65291		rejestr rastra. W systemie PAL występuje 312 linii, co wymaga rejestru 9 bitowego. Najstarszym bitem tego rejestru jest bit 0 rejestru 10. Dla ekranu 25-wierszowego używane są linie 4-203
12	FFOC	65292	0-1	bity 8-9 pozycji kursora (rejestr 13)
			2-7	nie używane
13	FFOD	65293	0-7	pozycja kursora sprzętowego
14	FFOE	65294		częstotliwość głos #1 (bit 0-7)
15	FFOF	65295		częstotliwość głos #2 (bit 0-7)
16	FF10	65296	0-1	częstotliwość głos #2 (bit 8-9)
			2-7	nie używane
17	FF11	65297'	0-3	głośność
			4	1=głos #1 włączony
			5	1=głos #2 włączony (ton)
			6	1=głos #2 włączony (szum)
			7	1=dźwięk wyłączony
18	FF12	65298	0-1	częstotliwość głos #1 (bit 8-9)
			2	bank ROM/RAM (0=RAM)
			3-5	adres pamięci obrazu HIRES (bit 13-15)
			6-7	nie używane
19	FF13	65299	0	clock status. 1=ROM bank
			1	„force single clock"-bit. (1=podwójna częstotliwość)
			2-7	adres generatora znaków (bity 10-15)
20	FF14	65300	0-2	nie używane
			3-7	adres pamięci obrazu i kolorów (bity 11-15)
21	FF15	65301		rej. barwy #0 bity 0-3=barwa, bity 4-6=jasność
22	FF16	65302		rej. barwy #1 bity 0-3=barwa, bity 4-6=jasność
23	FF17	65303		rej. barwy #2 bity 0-3=barwa, bity 4-6=jasność
24	FF18	65304		rej. barwy #3 bity 0-3=barwa, bity 4-6=jasność
25	FF19	65305		rej. barwy #4 bity 0-3=barwa, bity 4-6=jasność
26	FF1A	65306	0-1	bity 8-9 rejestru 27
			2-7	nie używane
27	FF1B	65307		rejestr „bit map reload"
28	FF1C	65308	0	bit 8 rejestru 29
			1-7	nie używane
29	FF1D	65309		aktualny wiersz rastra (bit 0-7)
30	FF1E	65310		aktualna kolumna rastra (bit 1-8)
31	FF1F	65311	0-2	„vertical subadresse" (8 linii rastra/wiersz)
62	FF3E	65342		ROM-select (zapis)
63	FF3F	65343		RAM-select (zapis).

Rejestry 62/63 nie znajdują się w układzie TED.

Sprostowanie

W majowym numerze naszego pisma, w artukule pt.: „Grzebanie w bajtach" wkradło się kilka błędów. Oto fragment listu autora artykułu:

„... w artykule znalazło się kilka przykrych błędów drukarskich, w tym trzy drobne (tzn. możliwe do skorygowania przez uważnego czytelnika) i jeden poważny. Zacznę od tego ostatniego. Na str. 7, u góry w prawej szpalcie, w akapicie rozpoczynającym omawianie rejestrów sterownika graficznego TED, zabrakło kilku wierszy, w efekcie czego z dwu instrukcji powodujących wyłączenie i ponowne włączenie ekranu ostała się tylko ta druga. Odpowiedni fragment powinien brzmieć:

Kolejna grupa „sztuczek" wiąże się z operowaniem rejestrami sterownika graficznego TED. I tak wykonując instrukcję

POKE 65286, PEEK(65286) AND 239

uzyskujemy efekt wygaszania ekranu (screen blanking), podobny jak przy ładowaniu programu z taśmy instrukcją LOAD. Przy wygaszonym ekranie szybkość przetwarzania komputera zwiększa się ok. 1.5-krotnie.

Instrukcja

POKE 65286, PEEK(65286) OR 16

włącza ekran ponownie.

Teraz kolej na błędy drobne: na początku artykułu, w ustępie omawiającym zawartość komórek 43-44, mylnie wydrukowano adres 4997. Na szczęście adres ten podany był równocześnie w układzie szesnastkowym ($1001), tak więc łatwo zauważyć i skorygować błąd. Podobnie łatwy do skorygowania błąd znalazł się u dołu prawej szpalty na str. 7 - znalazła się tam instrukcja o treści: „COLOR1, RCLRC(0), RLUMC(0)". Oczywiście ma być RLUM, a nie RLUMC - takiej funkcji w BASIC'u Commodore po prostu nie ma. Ostatnim błędem było pominięcie liczby $3000 na początku następnej strony tekstu, przy omawianiu adresów, pod którymi można umieścić w pamięci własną tablicę znaków: „Tablica znaków może być umieszczona tylko pod adresami stanowiącymi wielokrotność $400, czyli np. (12288), $3400 (13312), $3800 (14336), $3C00 (15360) itp." Brak wartości szesnastkowej przed „(12288)" wyraźnie rzuca się w oczy. [...]"

Wszystkich czytelników i autora - Jarosława Rafę przepraszamy za powstałe błędy.

Redakcja

Demo Scena

Muszę przyznać, że z prawdziwą przyjemnością wracam do dobrego pomysłu z czasów wydawania magazynu KEBAB na C64, jakim było publikowanie recenzji do programów demonstracyjnych, wykonanych w Polsce. Przez długi czas nic szczególnego w tej materii się nie działo, lecz jak nakazywała logika i doświadczenia wynikające z obserwacji scen komputerowych w innych krajach, należało spodziewać się i u nas rychłej zmiany sytuacji. No i doczekaliśmy się ! Otrzymałem dwa dema, które warte są obejrzenia! Mam szczerą nadzieję, że to początek rozkwitu polskiej sceny. Poświęćmy więc im kilka słów.

1. MegaDemo „Infosystem 91"

-programista (coder): Pampam (dawniej członek grupy DEUTER).

Demo zajmuje całą stronę dysku i składa się z 12 kolejnych części (nie licząc tzw. loader'ów, tj. części, które umilają ładowanie danych z dyskietki). Oglądając je możemy cieszyć oczy ładnymi grafikami (m.in. ładne liternictwo) i ciekawymi procedurami, które w polskich warunkach nie były do tej pory zbyt znane. Do takich niewątpliwie należy zaliczyć tzw. sinus scroll w którym poruszające się litery są miksowane z umieszczonym pod nimi obrazkiem czy też ciekawie animowane obiekty graficzne. A teraz kilka słów krytycznych pod adresem autora - po pierwsze demo razi trochę swoim niedopracowaniem (widać, że całość była składana w dużym pośpiechu), co jednak jest naturalne w przypadku pierwszych dzieł każdego programisty. Denerwował mnie również widoczny w niektórych miejscach brak znajomości podstaw języka angielskiego.

Ogólnie byłem jednak zaskoczony efektem pracy Pampama, czyli Pawła Goneta z Głogowa. Tak trzymać, Pawle !

2. Demo „Cruel Soldiers" grupy o tej samej nazwie

-programiści: Rekin i Ptysiu,

-grafika: Mr. Key i Hagar.

Od pewnego czasu miałem okazję śledzić rozwój stosunkowo „młodej stażem" grupy z Opola. Tym razem Okrutni Żołnierze nadesłali całkiem dojrzałą pracę. Na dysku program zajmuje jeden zbiór - przechodzenie z części do części odbywa się poprzez dekompresję fragmentu pamięci. Demo charakteryzuje się sympatyczną grafiką i interesującym coding'iem, zwłaszcza części z sinus scroll'em. Trudno znaleźć jakieś wyjątkowo uszczypliwe słowa krytyki - opisywany program demonstracyjny w zasadzie doskonale się mieści w jakościowych ramach odpowiednich dla produkcji tej wielkości. Zawarte w nim procedury nie mogą zachwycić, gdy porównamy je z najnowszymi trendami światowymi, ale trudno mieć o to pretensje.

Mam nadzieję, że opisywane powyżej dema są początkiem „radosnej twórczości" tego typu w Polsce.

Prosimy wszystkich o nadsyłanie do nas waszych prac-będą one wnikliwie analizowane, a najlepsze z nich opisywane i rozsyłane później na dyskietkach Public Domain.

Demonstracje prosimy nadsyłać na adres redakcji lub, jeszcze lepiej (szybciej dochodzą), na mój prywatny adres w przypadku C64 lub na adres Marcina Dudara (Duddie) w przypadku Amigi. Nasze adresy wszyscy piszący programy demonstracyjne z łatwością znajdą w naszych demach.

Uwaga. Zarówno megademo „Infosystem 91" jak i „Cruel Soldiers" ukazały się w majowym wydaniu C64+4 Public Domain Pack.

Paweł Sołtysiński (Polonus/Padua)

*** SLANG ***

BABOL - popularne określenie błędu popełnionego przez programistę.

BUFOR - miejsce w którym przechowuje się dane przeznaczone do wysłania w inny obszar.

BULLETS - podstawowa broń we wszelkich grach typu ZABIJ CO SIĘ RUSZA.

BUTTON - coś co się presi (patrz - Presi).

BYDGOSKI ZIELENIEC - popularne określenie dla najlepszego polskiego miesięcznika traktującego o komputerach firmy Commodore pt. "64 PLUS 4 & AMIGA"

BYTE - czyt. BAJT; jednostka informacji zawierająca osiem bitów; nazwa beznadziejnego zachodniego pisma którego 99% objętości zajmują reklamy a 1% wydruk ceny.

Suplement:

Preśnij batona: czynność oznaczająca naciskanie przycisku, najczęściej czerwonego, znajdującego się w tzw. joystick'u; pojęcie pochodzi od angielskich słów: PRESS BUTTON.

W ten sposób zakończyliśmy hasła zaczynające się na literę „B". W kolejnych numerach będziemy zajmować się hasłami na literę „D", bowiem jest ona kolejną literą w alfabecie. (????)

Sambor Kuźma

Music searcher 64

Program ten postanowiłem zadedykować wszystkim tym czytelnikom, którzy próbowali „wycinać" muzykę z programów, wykorzystując rady i wskazówki opublikowanymi w cyklu „Jak zrobić demo" w poprzednim miesiącu. Ograniczyłem się wówczas do podania paru podstawowych zasad szukania bloku muzyki w pamięci komputera, znajdującej się tam najczęściej jako element np. programu demonstracyjnego. Aby móc sprawnie odnajdywać blok zawierający procedurę grającą i dane, na podstawie których odtwarzana jest zarówno linia melodyczna i synteza, należy posiąść pewne informacje.

Do większości wniosków można dojść przez proste rozważania:

- Muzyka (np. w grach) pisana jest przez człowieka, który się w tej dziedzinie specjalizuje. Bardzo rzadko zdarza się, by muzykę pisał ten sam programista, który pisał samą grę (ludzi o tak wszechstronnych uzdolnieniach można policzyć na palcach jednej ręki).

- Muzyka pisana na zamówienie jest dostarczana jako samodzielna (tzn. pisana osobno) część pod postacią zwartego bloku danych. Taki blok jest potem umieszczany w przeznaczonym dla niego wcześniej miejscu, skąd później jest wywoływany przez program główny.

- Większość bardziej znanych muzyków używa specjalnie dla nich napisanych procedur muzycznych (ang. player) lub pisze z czasem własne (przykładem takiej sytuacji mogą być grupy muzyczne, takie jak np. Vibrants czy znana chyba wszystkim - Maniacs of Noise). Dobra procedura odtwarzająca muzykę powinna pozwalać na uzyskanie jaknajlepszej syntezy dźwięku i posiadać zdolność maksymalnego upakowania danych opisujących linię melodyczną (w formie krótkich sekwencji - ang. sequences lub patters).

- W dostarczonym bloku muzyki z punktu widzenia programisty, ważne są dwa adresy: pierwszy, ustawiający parametry (wskaźników na początek utworu, numeru utworu, o ile jest ich więcej) oraz drugi, który wskazuje na początek procedury, którą należy wywoływać podczas każdej obsługi przerwań, co zapewnia zachowanie równego tempa odtwarzania.

- Większość procedur grających jest wykorzystywana wielokrotnie, co pozwoliło na powstanie prezentowanego tutaj programu, rozpoznającego charakterystyczne fragmenty kodu maszynowego użytego w tych player'ach.

A teraz trochę o samym programie: pozwala on na błyskawiczne przeszukanie całej dostępnej pamięci komputera począwszy od adresu 4096 (1000 hex) aż do jej końca. Po rozpoznaniu jednej z 46 (!) znanych mu procedur grających, wyświetla on sposób jej uruchomienia a następnie zaczyna ją odtwarzać z jednoczesnym zaznaczeniem czasu, jaki jest poświęcony przez procesor na wykonanie wywołania player'a co każde przerwanie. Po wciśnięciu klawisza funkcyjnego F-1 odtwarzanie jest przerywane a program automatycznie przechodzi do poszukiwania następnych modułów muzycznych.

Tak więc aby posłużyć się naszym programem, należy w pierwszej kolejności wgrać do pamięci i uruchomić program, który zawiera interesującą nas muzykę, „zresetować" komputer w czasie, gdy gra muzyka a potem załadować „music searcher" i uruchomić go dyrektywą RUN. Jeżeli zostanie znaleziona któraś z procedur grających, wyświetlony zostanie porządek ustawienia rejestrów procesora i aktualny adres służący do ustawienia startu muzyki oraz adres (tzw. IRQ-driver), który należy wywoływać co każde przerwanie. Niektóre procedury (np.użyte w grach) posiadają zaprogramowanie kilku utworów, których wybór zależy od użytkównika.

Aby móc również ich posłuchać (o ile istnieją) można wcisnąć klawisze od 1 do 6 w czasie gdy „music searcher" odtwarza rozpoznaną muzykę. Niektóre z procedur grających podczas zmuszania ich do odtworzenia czegoś, czego nie ma, w najlepszym przypadku nie grają nic lub jakieś przypadkowe dźwięki a w najgorszym mogą zawiesić pracę procesora.

Prezentowany program został napisany w assemblerze, co gwarantowało jego dużą szybkość pracy oraz niewielki obszar zajmowanej przez niego pamięci.

Program ten znajduje się na naszym Public Domain Pack'u nr. 5. Będzie on również umieszczony na przygotowywanym taśmowym zestawie Public Domain - o szczegółach za miesiąc.

Paweł Sołtysiński.

JAK ZROBIĆ DEMO (3)

W listach napływających do redakcji zwracacie się z prośbą o zamieszczenie prostych procedur, których wykonanie sprawia często kłopot, a ich wyjaśnienia na próżno szukać w dostępnej w Polsce literaturze. Aby „wyjść na przeciw wymaganiom naszych Czytelników", postanowiłem w tej właśnie rubryce udzielć odpowiedzi na najczęściej pojawiające się pytania. Przejdźmy więc od razu do konkretów.

1. Jak wyświetlić na ekranie obrazek zrobiony na Advanced Art Studio ?

Jedno z najczęściej pojawiających się pytań. Przy odwzorowaniu zapisanych w pamięci danych w postaci graficznego obrazu w trybie „multicolor" (a takie obrazki generuje Advanced Art Studio) komputer potrzebuje trzech umieszczonych w RAM zbiorów, które zawierają informacje o:

a) wyglądzie ekranu graficznego (tzw. Bit-Map)

b) 1 zestawie atrybutów (do umieszczenia w obszarze aktualnego ekranu tekstowego),

c) 2 zestawie atrybutów (do umieszczenia w pamięci kolorów).

W przypadku standartowego ustawienia parametrów generowania obrazu (a więc stan po włączeniu komputera) wymienione wyżej dane powinny znaleźć się odpowiednio:

a) Bit-Map pomiędzy adresami $2000 a $3F40,

b) 1 zestaw atrybutów pomiędzy $0400 a $07E8 (normalnie: ekran tekstowy)

c) 2 zestaw atrybutów pomiędzy $D800 a $DBE8 (pamięć kolorów w C-64)

Po wgraniu gotowego obrazka (koniecznie rozkazem LOAD„nazwa", nr urządzenia, 1 lub z poziomu jakiegoś monitora) dane te znajdują się w następujących miejscach:

a) Bit-Map jest tam gdzie trzeba, tzn. od $2000 do 3F40;

b) 1 zestaw atrybutów leży od $3F40 do $4328;

c) 2 zestaw atrybutów leży od $4338 do $4720.

Po wygenerowanym przez Advanced Art Studio zbiorze zapisane są także dane dotyczące kolorów ramki i tła. Znaleźć je można pod adresem $4328 (kolor ramki) oraz pod $4329 (kolor tła). Nie pozostaje nam już nic innego jak tylko napisanie odpowiedniej procedury wyświetlającej gotowy obrazek:

```
      lda #$3B
      sta $DO11  ; włączamy tryb graficzny
      lda #$D8
      sta $DO16  ; zaznaczając, że ma to być 'multicolor'
      lda #$1E
      sta $DO18  ; teraz informacja, że Bit-Map leży
                   od adresu $2000
      lda $4328
      sta $DO20  ; ustawienie koloru ramki
      lda $4329
      sta $DO21  ; ustawienie koloru tła
      ldx #$00
loop  lda $3F40,X ; no i główna pętla...
      sta $0400,X
      lda $4040,X
      sta $0500,X
      lda $4140,X
      sta $0600,X
      lda $4240,X
      sta $0700,X
      lda $4338,X
      sta $D800,X
      lda $4438,X
      sta $D900,X
      lda $4538,X
      sta $DA00,X
      lda $4638,X
      sta $DB00,X
      inx
      bne loop
      rts        ; powrót z programu.
```

2. Jak wyciąć i uruchomić muzykę ?

Kolejne z pytań bumerangów. Odpowiedź na to pytanie nie może być ani jednoznaczna, ani wyczerpująca z tego względu, że procedury odtwarzające muzykę są niczym innym, jak tylko programami napisanymi w kodzie maszynowym i kwestie ich położenia w pamięci, itp. zależą od decyzji ich autora. Na szczęście wykrystalizowały się standardy, które znacznie ułatwią odpowiedź. Należy jednak pamiętać, że udzielone poniżej wskazówki odnoszą się do najczęściej spotykanych sytuacji, a więc najczęściej:

- „muzyki" zaczynają się od okrągłych adresów. Najczęściej od adresu $1000, $8000 i $E000;

- na ich początku znajdują się dwa lub trzy rozkazy JMP $adres. Może to wyglądać w ten sposób:

```
JMP $1041
JMP $107A
LDA #$12
STA $FB
...itd.
```

Pierwszy z tych JMP'ów może służyć do ustawiania muzyki od początku, drugi jest tzw. „driver'em" i należy go wywołać co każde przerwanie (aby muzyka grała). Czasami ich kolejność bywa odrotna...

- Najłatwiej jest zlokalizować muzykę szukając monitorem (rozkaz „h" -hunt) sekwencji bajtów: $8D, $18, $D4. Bajty te tworzą razem rozkaz STA $D418, czyli

ustawianie głośności w układzie dźwiękowym, co każda procedura grająca musi zrobić.

A jak już muzykę zlokalizowaliśmy, to możemy pokusić się o jej odtworzenie:

```
        sei
        lda #$35
        sta $01
        lda#$00
        jsr      'ustawienie muzyki od początku'
loop1   lda #$88
loop2   cmp $D012 ; czekanie na odświeżenie ekranu
        bne loop2
        jsr 'driver'
        lda $DC01
        cmp #$EF ; wciśnięto SPACE ?
        bne loop1 ; NIE — graj dalej
        lda #$37 ; TAK — powróć
        sta $01
        cli
        rts
```

3. Zrobiłem kilka modułów do kilkuczęściowego dema. jak przechodzić z jednej części do następnej?

Takich pytań nie było dużo, ale uważam, że już niedługo będzie to przydatne większej ilości osób.

Załóżmy, że przejście z aktualnej części do następnej odbywa się po wciśnięciu klawisza SPACE. Wynika z tego pierwszy wniosek, że klawisz musi być ciągle testowany, najlepiej w procedurze obsługi przerwań (regularne i częste testowanie).

Najlepiej uczyni to procedura:

```
        lda $DC01
        cmp #$EF   :wciśnięty SPACE?
        beq tak
        rts
tak     jmp relokator
```

Przedstawiona procedura sprawdza bezpośrednio port klawiatury, i po stwierdzeniu wciśnięcia SPACE (lub FIRE w porcie 1) skacze do procedury o nazwie „relokator", która zajmuje się wykonywaniem „zimnego restore" i po przesunięciu w dół następnej części dema — uruchomi ją.

A oto procedura „relocator":

```
        sei        :zimny restore
        jsr $FDA3
        jsr $FD15
        jsr $E518
        sei
        jmp relokuj
        .....
        .....
relokuj lda #$38
        sta $01
        lda #<położenie nowej części
        sta $FB
        lda # >położenie nowej części
        sta $FC
        lda #<adres docelowy
        sta $FD
        lda #a>dres docelowy
        sta $FE
        ldy #$00
loop    lda ($FB),y
        sta ($FD),y
        iny
        bne loop
        inc $FE
        inc $FC
        bne loop
        inc $FE
        inc $FC
        bne $01
        dec $01
        cli
        jmp start nowej części
```

Procedura „relokuj" została celowo oddzielona, ponieważ znajdować się musi w „bezpiecznym" miejscu pamięci, aby przypadkiem nie przerelokowała sama siebie... Zalecanym miejscem jej ulokowania jest adres $0340 (bufor magnetofonu).

Paweł Sołtysiński

ASSEMBLER 6510 lekcja 2

Cześć! Witam ponownie wszystkich fanów języka maszynowego i zapraszam do kolejnej lekcji. Tym razem zajmiemy się metodami adresowania pamięci.

Co to jest adresowanie? Rozkaz przesłania danych musi zawierać adres komórki pamięci, wskazywać rejestr lub podawać konkretną daną, której to przesłanie dotyczy. Podawanie takiego adresu nazywa się właśnie adresowaniem. Wszystkie tryby prześledzimy na przykładzie rozkazu LDA, który powoduje załadowanie danej do akumulatora.

TRYB NATYCHMIASTOWY

```
LDA #$64
```

Po wykonaniu tej instrukcji w akumulatorze znajdzie się wartość 64. Widać więc, że trybu tego używa się do ładowania danych bezpośrednich. Łatwo go odróżnić od

innych trybów gdyż przed wartością ładowaną znajduje się znak „#".

TRYB BEZPOŚREDNI NA STRONIE ZEROWEJ

```
LDA $01
```

Brak # przed liczbą oznacza, że do akumulatora zostanie załadowana wartość znajdująca się pod adresem wskazywanym przez tą liczbę. Jeśli np. w komórce pamięci o adresie $01 znajduje się wartość 37 to po wykonaniu naszego przykładu w akumulatorze pojawi się liczba 37.

TRYB BEZPOŚREDNI

```
LDA $0400
```

Tryb ten działa tak samo jak poprzedni z tą różnicą, że pozwala na zaadresowanie całej pamięci komputera (64 KB).

TRYB INDEKSOWY NA STRONIE ZEROWEJ

```
LDA $40,X
```

Po wykonaniu takiej instrukcji w akumulatorze znajdzie się zawartość komórki o adresie $40+X. Możliwy jest również taki zapis:

```
LDX $10,Y
```

po kórym w rejestrze X zostanie umieszczona wartość znajdująca się w komórce o adresie $10+Y.

TRYB INDEKSOWY X I INDEKSOWY Y

```
LDA $0400,X
LDA $1000,Y
```

Tryb ten zachowuje się tak samo jak poprzedni lecz pozwala na adresowanie całej pamięci.

TRYB POŚREDNI POSTINDEKSOWANY

```
LDA ($FB),Y
```

Dwie komórki FB i FC tworzą adres bazowy rozkazu (FB - bajt młodszy, FC - bajt starszy). Do tego jest dodawana wartość rejestru Y i z tak obliczonego adresu jest pobierana dana, np. jeśli komórka FB=$E2, FC=$10 i Y=1 to dana zostanie pobrana z adresu $10E3.

TRYB POŚREDNIU PREINDEKSOWANY

```
LDA ($10,X)
```

Adres właściwy znajduje się w dwu kolejnych komórkach, z których pierwsza ma adres $10+X. Jeśli np. 11=$E2, 12=$10 i X=1 to po wykonaniu LDA ($10,X) w akumulatorze znajdzie się zawartość komórki o adresie $10E2.

To już wszystkie tryby adresowania dostępne w procesorze 6510. Aby lepiej je poznać i dojść do wprawy w ich używaniu trzeba po prostu napisać kilkanaście prostych programików wykorzystujących każdy z nich. Poniżej podaję prościutki relokator czyli program (a właściwie procedurę), która przenosi jeden obszar pamięci w inny. Przeanalizujcie ją i spróbujcie napisać własne relokatory wykorzystujące inne metody. Zastanówcie się jak przerelokować większą ilość pamięci niż $FF (256 bajtów).

```
1000    LDA #$00
        LDX #$04
        STA $FB
        STA $FD
        STX $FC
        INX
        STX $FE
        LDY #$00
100F    LDA ($FB),Y
        STA ($FD),Y
        INY
        BNE $100F
        BRK
```

Miłego „klepania" w klawisze życzy:

Sambor Kuźma

Graj aby wygrać

Slap Fight

POKE 2879,ilość żyć
POKE 10239,173 -nieśmiertelność
Start: POKE 2048,2
POKE 2049,56
POKE 2050,253
SYS 31488

Dan Dare

POKE 23603,76
POKE 23604,88
POKE 23605,92 energia i czas nieskończone
SYS 16384

Bounder

Na ekranie tytułowym nacisnąć jednocześnie klawisze: 1 Q A Z „spacja".

Marauder

Aby wyłączyć kolizję należy nacisnąć klawisze: C= Q 2 'spacja'

Cybernoid

POKE 28870,169 nieskończone życie
POKE 23317,234
POKE 23318,234 nieskończona broń
SYS 2084

Nieśmiertelność można również uzyskać definiując klawisze jako:

Y X E S (SEXY)

Pro BMX Simulator

POKE 8734,0 czas
SYS 4096

IO - Game

POKE 42348,0 kolizja
POKE 25117,189 życie
Start: POKE 32768,32
SYS 24586

Aliens (wersja dyskowa)

Aby przeskoczyć do następnego poziomu wystarczy nacisnąć:

* | RESTORE

Tarzan

POKE 47226,234
POKE 47227,234
POKE 47228,234
POKE 52848,234
POKE 52849,234
POKE 52850,234 życie
SYS 294

Arkanoid

Włącz grę na opcję dla dwóch graczy i strać wszystkie „życia" dla pierwszego licznika. Po chwili za każdą uderzoną cegiełkę komputer będzie zwiększał ilość „żyć"

S.K.

JEWELS OF DARKNESS

Jaskinia Olbrzymia to imię Mocy, szeptane w ciemnych zaułkach, uosabiające niewyobrażalne bogactwo i niewysłowioną grozę. Rozsądni ludzie twierdzą, że jest to mit, produkt rozgorączkowanej wyobraźni - lecz...lecz każdego roku wielu ludzi wyprawia się na jej poszukiwanie, kierowani biedą lub ogarniającą ich umysły żądzą.

Niektórzy z nich wracają jedynie z opowieściami o bezowocnych poszukiwaniach lub z kłamstwami o ogromnych niebezpieczeństwach, które stoją na drodze do jaskini: wulkanach, smokach, gigantycznych wężach i innych strasznych stworach z piekła rodem. Lecz wielu nie wraca... O tych mówi się, że znaleźli jaskinię i zginęli lub może, korzystając z ogromnego majątku tam znalezionego, przenieśli się do dalekich krajów gdzie żyją w bogactwie i luksusie. To był niesamowity przypadek, że siedziałeś akurat w tawernie kiedy wszedł zmęczony drogą wojownik i zamawiając piwo zapłacił wysokiej wartości starą monetą twierdząc jednocześnie, że był w Jaskini Olbrzymiej i uszedł z życiem!

Nikt mu nie uwierzył. „Złodziej" i „najemnik" - tak sobie wyjaśniono jego bogactwo lecz ty nie byłeś taki pewny. Przybysz wyglądał na prawego człowieka i odmawiał jakichkolwiek wyjaśnień na temat swoich odkryć. Kiedy uzmysłowiłeś sobie, że wieśniacy zaplanowali zasadzkę mającą na celu pozbawienie przybysza pieniędzy (i przy okazji życia) wyprowadziłeś go bezpiecznie tylnym wyjściem.

„Stokrotne dzięki!" zawołał kiedy dźwięki pogoni ostatecznie ucichłyi. „Niech ci bogowie wynagrodzą twój uczynek ! Obawiam się jednak, że twoi przyjaciele mogą chcieć się zemścić jeżeli tylko pokażesz się w mieście."

„Aby wynagrodzić ci twoje męstwo" przybysz ciągnął „ofiaruje ci najwartościowszą rzecz jaką posiadam... plan dotarcia do Jaskini Olbrzymiej" i szybkim ruchem ręki wcisnął ci w dłoń zmięty kawałek papieru. Szczerze mówiąc spodziewałeś się bardziej konwencjonalnej zapłaty i to dość dużej, lecz mężczyzna wyglądał na dobrego w walce więc zaakceptowałeś skromną nagrodę i nawet zdobyłeś się na szczery uścisk dłoni.

Wbrew swoim podejrzeniom, że mapa jest fałszywa, postanowiłeś wyruszyć w podróż. Na pewno pomógł ci w tym strach przed rozgniewanymi opryszkami z karczmy. I nagle... okazało się, że mapa jest dobra! Prowadziła cię przez niezamieszkałe tereny: góry, lasy, pustynie. Aż pewnego dnia zaprowadziła cię do Jaskini Olbrzymiej. Przynajmniej taką miałeś nadzieję...

I wtedy to się stało! Opierałeś się o ścianę małego, ceglanego budyneczku, otoczonego gęstym lasem, spod którego wypływała rzeczka płynąca dalej na południe. Na północ rozpościerała się otwarta równina ginąca we mgle. Wyjąłeś mapę aby wytyczyć dalszą marszrutę gdy nagły poryw wiatru wyrwał ci ją z ręki i wrzucił prosto do wody. Ratunek nie był możliwy. Bystry strumień porwał ją błyskawicznie, zresztą i tak nie umiałeś pływać.

Teraz tylko od ciebie zależy czy znajdziesz jaskinie i zdobędziesz ogromne bogactwa.

Stjąc tak i myśląc przypominasz sobie nagryzmoloną notatkę na marginesie mapy:

„Uwaga. W jaskini działa Magia"

To co przeczytaliście to tylko legenda do pierwszej części gry „Jewels Of Darkness". Druga mówi o walce dobra i zła, o twojej walce z Panem Ciemności Agaliarept'em. Trzecią misją jest zdobycie ogromnych bogactw pokonanego Agaliarept'a.

Gra jest naprawdę bardzo dobrze zrobiona. Na uwagę zasługuje zwłaszcza świetny interpreter zdań, który pozwala nawet na wpisanie takich sekwencji:

„Examine all but the cross, dragon and lamp and go east"

Autorzy zainstalowali bardzo pomocne funkcje RAM SAVE i RAM LOAD, które pozwalają zapisać stan gry w pamięci RAM. Opcja DOPS cofa jeden ruch w tył! W ten sposób wszystkie popełnione błędy można szybko i łatwo naprawić.

Jedyną wadą gry jest duża ilość lokacji (ok. 200).

Czasem trzeba się sporo nachodzić aby trafić w odpowiednie miejsce.

„Jewels Of Darkness"

Koncepcja i teksty:	Pete Austin
Programowanie:	Mike, Nick i Pete Austin
Grafika:	James Horsler
Wydawca:	Rainbird Software
Komputer:	Commodore 64
Ocena:	
Grafika:	3
Muzyka:	–
Animacja:	–
Pomysł:	6
Przyjemność grania:	6
Ogólnie:	5.6

Hi-Man

Co znaczy oryginał

Polska jest wyjątkowym krajem. Nigdzie na świecie nie jest tak rozwinięte złodziejstwo jak właśnie w Polsce. W żadnym cywilizowanym kraju legalna, zarejestrowana firma nie może uprawiać piractwa. U nas może!

Problem piractwa był już wielokrotnie poruszany na łamach wielu pism, nie tylko komputerowych. Chciałbym poruszyć go jeszcze raz, tyle, że z nieco innej strony.

Myślę, że 100% posiadaczy Amigi czy C-64 zaopatruje się w programy u giełdziarzy, różnego rodzaju pseudo firm softwarewych lub drogą wymiany. Większa część użytkowników komputerów nie ma pojęcia, jak wygląda oryginalny program sprzedawany przez firmę w normalnym sklepie. Szkoda, jest bowiem na co popatrzeć.

Leży teraz przede mną oryginalna gra „Jewels of Darkness". Przyjemna, kolorowa okładka już mówi, że warto było za nią zapłacić. Otwieram pudełko. W środku znajduję kilka kartek - są to instrukcje ładowania gry w kilkunastu językach. Wyjmuje je i odkładam na bok. Teraz znajduję dość grubą (32 strony) książeczkę wydaną na pięknym, kredowym papierze. Co jest w środku? Na początku dokładna instrukcja ładowania. Nawet najbardziej początkujący użytkownik komputera poradzi sobie z uruchomieniem gry. Przekładam kartki i widzę legendy do każdej z części gry. Dalej jest dokładna instrukcja używania słownika, jest to bowiem gra przygodowa. Na końcu książki zamieszczono świetne opowiadanie z gatunku fantasy, na podstawie którego napisano scenariusz gry. Teraz już nie będę musiał spędzić kilku godzin na rozgryzaniu „o co tu chodzi? "

To jednak nie wszystko. Pod książeczką leży kartka zatytułowana „Hint Request". Co to jest? Otóż firma wychodzi na przeciw zapotrzebowaniu graczy na różnego rodzaju podpowiedzi do swoich gier. Jeśli ktoś gdzieś „utknął" wystarczy wypełnić formularz i wysłać na adres firmy. W prezencie dostaje się wtedy darmowe podpowiedzi do gry. Prawda, że piękne?

Na samym dole pudełka leży elegancko zapakowana dyskietka. Wystarczy ją włożyć do stacji, by przenieść się do świata stworzonego przez programistów.

Jak widzicie, warto mieć oryginalne gry. Do wielu innych tytułów często dodawane są książki, np. gra „Lord of The Rings" była sprzedawana z książką Tolkiena o tym samym tytule. Do sławnej już gry „The Last Ninja" firma dodawała oryginalny strój Ninja, miecz i shurian'y.

Nie są to jednak główne zalety posiadania oryginału. Najważniejsze jest to, że mamy 100% pewności sprawności gry. Ileż to razy zdarzało się, że pozornie dobra gra po pewnym czasie „sypała się". Wydane na nią pieniądze przepadają wtedy bez powrotu w przepastnych kieszeniach giełdowych handlarzy.

Gry są tylko jednak częścią oprogramowania. Co jednak z programami użytkowymi. Tu problem jest poważniejszy.

Większość firm tworzących oprogramowanie użytkowe specjalnie nie zamieszcza instrukcji i podpowiedzi w środku programu, aby delikwent który nabył piracką kopię był i tak zmuszony do kupienia oryginału. Stąd w listach do redakcji pojawiają się prośby typu: „Jak używać Virus Expert'a?" lub „Ludzie ratunku! Nie wiem jak posługiwać się Sculpt'em 4D!".

W kilku programach znalazłem hasło, które moim zdaniem trafia w sedno problemu:

A program worth using is worth buying!

Program wart użycia jest wart nabycia!

Sambor Kuźma

p.s.

W związku z dużym zapotrzebowaniem na instrukcje do programów użytkowych będziemy starali się je drukować w naszym piśmie, oczywiście w znacznie skromniejszej niż oryginały, formie. Już rozpoczął się cykl poświęcony Sculpt'owi 4D, a w tym numerze znajdziecie opis Virus Expert'a. Następnym programem będzie Disk Master. Czekamy również na Wasze propozycje.

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace.

Tematyka programów dowolna.

Konkurs rozgrywany jest osobno dla komputerów C-16 i C-64.

Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane).

Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu.

Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po **500.000** zł każda. Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma.

Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

Programowanie w AMOS Basic'u (cz.2)

W poprzednim odcinku zajmowaliśmy się strukturalnymi instrukcjami cyklu oraz innymi rozkazami sterującymi. Dziś zajmiemy się procedurami Amos'a oraz zmiennymi.

Jednak najpierw errata do poprzeniego odcinka. Przez nieuwagę wadliwie została podana składnia instrukcji IF. Otóż IF w Amosie posiada dwie postacie. Pierwsza naśladuje standardowy Basic i ma postać:

```
IF warunek THEN komenda_1: komenda_2: [...]:
komenda_n
```

gdzie „warunek" jest wyrażeniem warunkowym lub numerycznym zaś po „THEN" nastepuje lista komend do wykonania jeśli warunek jest spełniony. Druga postać instrukcji bardziej nawiązuje do Pascal'a niż do Basic'a:

```
IF warunek
      komenda_1
      ...
      komenda_n
ELSE
      komenda_1
      ...
      komenda_m
ENDIF
```

Jej działanie opisane zostało w poprzednim artykule. Warto zwrócić uwagę na brak 'THEN' w tej postaci instrukcji 'IF'

SKOKI I PODPROGRAMY

Instrukcje cyklu i warunkowa nie wyczerpują bogatego zestawu instrukcji sterujących AMOS'a.
W AMOS'ie zaimplementowane zostały instrukcja skoku GOTO oraz wywołania podprogramu GOSUB. Obie znane są doskonale ze standardowego BASIC'a. Istnieje jednak pewna różnica - w AMOS'ie nie ma numerów linii do których można by skoczyć! Istnieje zatem potrzeba oznaczania pozycji w tekście programu w inny sposób. Służą do tego tzw. etykiety. Etykieta to napis w tekście programu zakończony dwukropkiem i znajdujący się jako pierwsza komenda w linii. Przykładowo oznaczamy linię etykietą 'START':

```
START: Print "Program uruchamia się po skoku do
tego  miejsca"
```

Po umieszczeniu etykiety w tekście możemy się do niej odwoływać przez jej nazwę. Etykiety są niezbędne do działania instrukcji 'GOTO'. Jej postać jest następujca:

```
GOTO    nazwa_etykiety
```

Powoduje to skok (przekazanie sterowania) do miejsca oznaczonego etykietą lub bezpośrednio za nią jeśli w linii etykiety nie ma innych rozkazów.

Przykład:

```
skok:
Print "ta pętla nigdy się nie kończy!"
Goto skok
```

Podobną budowę ma instrukcja GOSUB (Go Subroutine - skocz do podprogramu). Powoduje ona podobnie jak GOTO, skok do etykiety lecz posiada także inną pożyteczną właściwość. Jest to bowiem tzw. skok ze śladem tj. zawsze zapamiętane jest miejsce z którego został wykonany skok. Podprogram powinien kończyć się instrukcją RETURN, która powoduje powrót sterowania do instrukcji umieszczonej bezpośrednio za instrukcj GOSUB. Dokładny opis wykorzystania instrukcji podprogramu wykracza poza ramy tego opisu. Przytoczmy jedynie przykład:

```
Rem  Program oblicza pierwiastki
     wprowadzonych   liczb
Rem  nie pozwala na wprowadzanie liczb
     ujemnych
Rem  wykorzystuje do wprowadzania liczb podprogram

Rem
Do
     GoSub WczytajA
     Print "Pierwiastek ";a;" Wynosi ";sqr(a)
Loop WczytajA:
Rem  Podprogram wczytujący do A liczbę dodatnią
Wczyt:
     Input "Podaj liczbę",a
If a Then Goto Wczyt
Return
```

Jeszcze jedna uwaga. Z pewnością wszyscy znają instrukcję REM lecz dla pewności opiszemy ją .

REM (Remark - komentarz) służy do komentowania fragmentów programu. Zarówno ona sama jak i następujący po niej tekst jest całkowicie ignorowany przez interpreter BASIC'a. Uwaga, w AMOS Basic'u możemy użyć znaku ' zamiast REM (Uwaga! znak ' różni się od znaku apostrofu '). Dla słabo zorientowanych w klawiaturze podaję informację, że odpowiedni klawisz znajduje się pod ESC.

Jeśli chcemy zakończyć działanie podprogramu przed dotarciem do RETURN to możemy tego dokonać instrukcją POP.

Komendę POP możemy interpretować jako skok do miejsca programu w którym znajduje się RETURN.

W gruncie rzeczy instrukcja GOSUB jest umieszczona w AMOS'ie głównie dla kompatybilności z innymi implememntacjami BASIC'a. Użycie GOSUB jest zbędne jeśli posługujemy się procedurami (poza naprawdę nielicznymi przypadkami).

Z podobnego względu w AMOS'ie można też numerować linie! Numery tym si różnią od etykiet, że nie muszą być zakończone dwukropkiem. Tak więc można w AMOS Basicu przepisywać przykłady z większości BASIC'ów bez najmniejszej zmiany!

PROCEDURY

Wspomniane już procedury są niezwykle silnym i wydajnym mechanizmem pisania programu.

Procedury to specjalnie zbudowane programy pełniące funkcje „czarnych skrzynek". Definiowanie procedur przez programistę tworzy niejako nowe rozkazy AMOS'a. Do uruchomienia procedury wystarczy podać jej nazwę i ewentualnie w nawiasach prostokątnych listy parametrów. Zasadniczo procedury komunikują się z programem przez parametry. Wnętrze procedury jest jakby niezależnym programem. Zazwyczaj nie są widziane tutaj zmienne programu wywołującego poza parametrami. Możliwe jest jednak wymuszenie współdzielenia niektórych zmiennych programu przez procedurę i program główny. Aby nie zaplątać się w teoretyczne rozważania przystąpmy bezpośrednio do opisywania mechanizmów związanych z procedurami.

Definicję procedury możemy umieścić w dowolnym miejscu tekstu programu. Dobrym zwyczajem jest umieszczanie wszystkich procedur w jednym miejscu np. na końcu programu. Treść procedury jest „omijana" przez BASIC w czasie interpretacji programu. Do używania procedury wystarczy w treści programu umieścić jej definicj - nie potrzeba wykonywać żadnych dodatkowych czynności. Budowa definicji procedury jest usystymatyzowana w następujący sposób:

1) Rozpoczyna się od nagłówka „Procedure" po którym następuje nazwa procedury.

2) Po nazwie opcjonalnie mogą występować parametry umieszczone w nawiasach prostokątynych i oddzielone przecinkami.

3) Po nagłówku występuje treść procedury będąca dowolym ciągiem instrukcji AMOS BASIC'a.

4) Procedura zawsze musi kończyć się instrukcją END PROC.

A oto schematyczny zapis:

```
Procedure Nazwa_procedury [p1 , ... , pn]
      i1
      i2
      ..
      in
End  Proc
```

Przykładowo napiszemy procedurę drukującą na ekranie n razy podany tekst:

```
Procedure Druk [n,t$]
      Rem
      Rem Drukuje n razy T$
      Rem
      for z=1 to n
      Print t$
      next
End  Proc
```

Zauważcie, że procedura „Druk" używa własnej zmiennej Z. Jest to zmienna całkowicie niezależna od głównego programu. Jeśli zmienna o nazwie Z jest używana w głównym programie to użycie procedury w żadnym stopniu jej nie zmienia.

Zmienna Z jest tworzona przy pierwszym jej użyciu w tekscie i niszczona po napotkaniu END PROC. Jeśli teraz chcesz użyć ciekawej cechy edytora AMOS'a to naprowadź kursor na procedurę i wciśnij klawisz F9. Tak! Procedura zwija się do swojego nagłówka.

Procedurę możemy użyć przez wpisanie jej nazwy wraz z parametrami. Możemy więc do tekstu napisanej już procedury dopisać (najlepiej przed nią):

```
Rem  Demonstracja użycia procedury
ile=3
tekst$="AMOS"
Druk[2,"Drukujemy"]
Druk[n,tekst$]
Druk[n+2,tekst$+" BASIC"]
Rem  Tu umieść treść procedury
```

Jak widzimy jako parametr procedury można podać stałą, zmienną lub wyrażenie. Między innymi dlatego zmiana parametru w procedurze nie jest widziana na zewnątrz programu. Tak więc zmienne N i T$ są jedynie tworzone na użytek podprogramu i tylko tam widziane.

Procedury możemy bardzo efektywnie używać do wspomagania projektowania programu. Wystarczy stwierdzić, że program będzie składał się z trzech części: inicjującej, programu głównego i zakończenia. Traktujemy te trzy części jako niezależne i piszemy program:

```
Rem  Duży program
Inicjacja
Prg_Glowny
Koniec
Stop
Procedure Inicjacja
End Proc
Procedura Prg_Glowny
End Proc
Procedure Koniec
End  Proc
```

Oczywiście napis „Inicjacja" to wywołanie procedury nie posiadającej parametrów. Ponieważ każda z napisanych procedur nie posiada żadnych instrukcji zatem uruchomienie tego szkieletu programu klawiszem F1 nie przyniesie żadnego efektu.

Teraz możemy stopniowo dopisywać fragmenty programu aż do osiągnięcia zadowalającego efektu. W napisanych już procedurach możemy umieszczać wywołania następnych procedur itd. (Uwaga definicje procedur nie mogą być umieszczone w innych procedurach).

ZASIĘG ZMIENNYCH

Aby zrozumieć dokładnie operowanie procedur na zmiennych należy zaznajomić się z mechanizmem ich zasięgu i przesłaniania. Dzięki temu można uniknąć wielu trudnych do lokalizacji błędów i w pełni wykorzystać cechy języka.

BASIC przyjmuje prostą konwencję operowania zmiennymi. Pierwsze użycie danej zmiennej powoduje jej utworzenie i wyzerowanie. Pozwala to na nieużywanie definicji zmiennych. Pomyłkowe użycie zmiennej (np. przez drobną zmianę nazwy) prowadzi najczęściej do trudnych do wykrycia błędów. Zmienna raz utworzona istnieje aż do zakończenia programu. Pisanie dużych programów w standardowym BASIC'u jest szczególnie trudne ponieważ cały czas należy uważać czy pomyłkowo nie wykorzystujemy zajętej już nazwy zmiennej. Pisanie podprogramów niezależnych od otoczenia w którym powstały jest faktycznie niemożliwe.

Wprowadzenie procedur do AMOS'a wymaga nowego podejścia do zmiennych. Po pierwsze procedura jest autonomiczną częścią programu. Po uruchomieniu procedury staje się ona „ślepa" na wszystkie zmienne programu który je wywoła . Próba odczytania zmiennych programu głównego kończy się otrzymaniem zera lub tekstu pustego. Zapewnia to całkowite bezpieczeństwo użycia zmiennych w procedurze. Nie musimy się martwić o kolizję nazw zmiennych między procedurą i głównym programem. Sprawia to, że te same procedury mogą najczęściej bezpośrednio być używane w różnych programach. Wszystkie zmienne utworzone w czasie działania procedury (np. liczniki pętli) są kasowane po wyjściu z procedury. Tak więc zasięgiem zmiennej utworzonej w programie głównym jest program główny zaś zasięgiem zmiennej utworzonej w procedurze jest procedura.

Często pojawia się potrzeba aby procedura wyliczyła jakąś wartość i zwróciła ją w zmiennej możliwej do odczytania przez program główny. Tu można posłużyć się trzema sposobami: zwracanym parametrem, zmienną współdzieloną lub zmienną globalną.

Metoda pierwsza - przekazywany parametr.

Aby programowi głównemu udostępnić wyliczoną wartość należy użyć w specjalny sposób komendy END PROC: END PROC [wartość_wyliczona]. Następnie w programie głównym odczytać ją przy pomocy instrukcji umieszczonej bezpośrednio za wywołaniem procedury: wartość=Param. Należy zauważyć, że zmienna „wartość" jest zmienną programu głównego zaś zmienna „wartość-wyliczona" zmienną należącą do procedury. Jeśli ma zostać zwrócony parametr typu tekstowego należy analogicznie użyć instrukcji Param$.

Metoda druga - zmienna współdzielona.

Użycie instrukcji: Shared nazwa_zmiennej w tekście procedury powoduje, że zmienna programu głównego jest widziana w procedurze. Wszelkie jej zmiany dokonane w czasie działania procedury będą widoczne w programie głównym. Metoda ta jest mniej uniwersalna niż pierwsza gdyż należy jedną (lub więcej) zmienną programu głównego przeznaczyć na współpracę z procedurą. Ma jednak tę zaletę że procedura może zwracać efekty swej działalności w kilku zmiennych. Uwaga! Jeśli nazwa zmiennej typu shared koliduje z nazwą parametru procedury to ta pierwsza ma większy priorytet.

Metoda trzecia - zmienna globalna.

Użycie instrukcji: Global nazwa_zmiennej w tekście programu głównego powoduje, że zmienna ta będzie współdzielona z wszystkimi procedurami. Jest to mechanizm szczególnie niebezpieczny - każda procedura może dokonać zmiany tej zmiennej. Właściwie używa się tej metody do niewielkiej klasy zmiennych, które powinny być widziane w wielu procedurach.

Przykład:

```
Global a
a=3
b=4
c=5
p[b]
Print  param

Procedure P[z]
Shared c
       Print z
       Print a
       Print c
End    Proc[z+10]
```

Program ten wyprowadza liczby: 4,3,5,14

Na koniec opisu funkcjonowania procedur warto wspomnieć o instrukcji podobnej do POP (używanej razem z GOSUB). W dowolnej chwili można przerwać działanie procedury wychodząc poziom wyżej przy pomocy instrukcji POP PROC.

Tyle o procedurach i sterowaniu. Następne odcinki będą opisywały obsługę grafiki i animacji.

Na koniec krótkie przypomnienie poznanych instrukcji:

Instrukcje cyklu:
For-Step-Next
While-Wend
Repeat-Until
Do-Loop Instrukcja rozgałęzienia warunkowego:
If-Then-Else-Endif Instrukcje skoku i skoku z rozgałęzieniem
Goto
GoSub,Pop,Return Procedury
Procedure-End Proc
Pop Proc
Shared,Global
Param,Param$

Czekamy na Wasze opinie o tym cyklu artykułów. Być może jest zbyt łatwy, może za trudny? Jeśli macie jakiekolwiek pytania bądź wątpliwości - piszcie! Będziemy starali się zamieszczać odpowiedzi na Wasze listy.

Krzysztof Moroł

Obesse Fat Agnus 1 MB

Instalowany obecnie w Amigach nowy Agnus (Obesse Fat Agnus) może obsługiwać 1 MB pamięci Chip. Nie było by w tym nic nienormalnego, ale – jeżeli posiadamy 0.5 MB pamięci Chip (od adresu 0 do$7FFFF), oraz 0.5 MB pamięci Fast (od adresu $C0000 do $CFFFF), to pamięć typu Fast jest widziana przez Agnusa pod adresem od $80000 do $FFFFF. Oznacza to, że możemy umieszczać w niej ekrany, sample, Cooper List'y oraz bufory stacji dysków. Jeżeli np. załadujemy sampla od adresu $C57800, to możemy go odtworzyć, pamiętając jednak, by uruchomić go od adresu $d7800.

A teraz na podstawie magazynu dyskowego grupy D-Tect „Hack-Mag" przytoczę nazwy i adresy nowych rejestrów w Obesse Fat Agnus:

$05A - BLTCON2	$05B - BLTCON01
$05c - BLTSIZV	$05E - BLTSIZH
$078 - SPRHDAT	$07C - DENISEID
$106 - BPLCON3	$1C0 - HTOTAL
$1C2 - HSSTOP	$1C4 - HBSTRT
$1C6 - HBSTOP	$1C8 - VTOTAL
$1CA - VSSTOP	$1CC - VBSTRT
$1CE - VBSTOP	$1D0 - SPRHSTRT
$1D2 - SPRHSTOP	$1D4 - BPLHSTRT
$1D6 - BPLHSTOP	$1D8 - HHPOSW
$1DA - HHPOSR	$1DC - BEAMCON0
$1DE - HSSTRT	$1E0 - VSSTRT
$1E2 - HCENTER	$1E4 - DIWHIGH
$1E6 - BPLHMOD	$1E8 - SPRHPTH
$1EA - SPRHPLT	$1EC - BPL1HPTH
$1EE - BPL1HPTL	

Komórka BEAMCON0 ($DFF1DC) odpowiada za tryb wyświetlania obrazu (zmieniając bit 5 z zapalonego na zgaszony przechodzimy z trybu PAL na NTSC).

Niestety, ani ja, ani autorzy „Hack-Mag" nie znamy przeznaczenia podanych rejestrów. Jeżeli ktoś z Czytelników ma opis tych rejestrów oraz bitów w tych rejestrach, to mógłby podzielić się wiadomościami z pozostałymi użytkownikami Amigi oraz z redakcją.

Marcin „Duddie" Dudar

Kopiowanie długich zbiorów

Używając normalnego programu kopiującego zauważyliście zapewne, że korzysta on z RAM-disk'u. Jest to bardzo nieefektywny sposób kopiowania. W praktyce, na Amidze z 0.5 MB, nawet jeśli uzyskamy (po zabiciu jednego bitteplane'a i zamknięciu okna) około 300 kB wolnego RAM'u, to kłopoty pojawią się już przy próbie skopiowania zbioru długości ponad 100 kB! Dzieje się tak dlatego, że RAM-disk jest rozrzucony po całej pamięci komputera i nie tworzy jednego ciągu, potrzebnego do skopiowania długiego zbioru.

Jak sobie z tym poradzić? Odpowiedź jest prosta! Należy użyć monitora języka maszynowego. Ja pracuję na wersji 1.7 D-Mon. Wy go pewnie nie macie, ale do naszych celów nadają się również inne monitory, np. Timo Rossi Monitor.

Dla uzyskania jak największego wolnego obszaru pamięci, po „zabutowaniu" dysku trzeba uruchomić program „Add 21k". Spowoduje to usunięcie jednego koloru z ekranu. Należy również zmniejszyć okienko DOS'u (zawsze to kilka bajtów więcej). Jeśli macie podłączoną do komputera dodatkową stację dysków, to należy ją wyłączyć (jej bufor nie będzie wówczas zajmował miejsca w pamięci komputera). Do tak przygotowanej Amigi wczytujemy monitor. Teraz ilość wolnej pamięci zależeć będzie od długości samego monitora. Im monitor krótszy, tym więcej pamięci zastanie dla użytkownika. Z pewnym zapasem można uznać, że od adresu $25000 zaczyna się wolny obszr dostępny dla naszych celów. Możemy więc kopiować zbiory o długości aż 372735 bajtów!!! Jeżeli zamiast D-Mon'a użyjecie Timo Rossi Monitor, to programy możecie ładować od adresu $20000, a więc długość kopiowanych zbiorów może wynosić aż 393216 bajtów!

Na czym polega kopiowanie? Na wczytaniu do pamięci komputera danego zbiory, a potem zapisaniu go na innym dysku. Rozkaz LOAD w monitorze zastępowany jest przez znaczek '['. Piszemy więc:

```
[ 'nazwa' adres , np.:
[ 'ik+' 25000          dla D-Mon, lub
[ 20000 ik+            dla TRM
```

Obydwa monitory podadzą nam dane o załadowaniu bloku. Teraz następuje najtrudniejsza część operacji. Monitory mają różnie zorganizowane składnie rozkazu SAVE. W D-Mon'ie po adresie startowym (w naszym przykładzie $25000) podaje się **adres końca zbioru.** W TRM po adresie początkowym podaje się **długość zbioru!** Należy bardzo uważać, by nie popełnić pomyłki, gdyż spowoduje to niedziałanie programu. W niektórych wersjach TRM opcja SAVE działa tak samo jak w D-Mon'ie. Aby się upewnić co do składni rozkazu SAVE, należy zajrzeć do Help-Menu (klawisz HELP) i poszukać rozkazu ']'. Zapis:

```
] addr1 addr2 name
```

oznacza, że należy podać adres początku i końca zbioru, natomiast

```
] addr lngt name
```

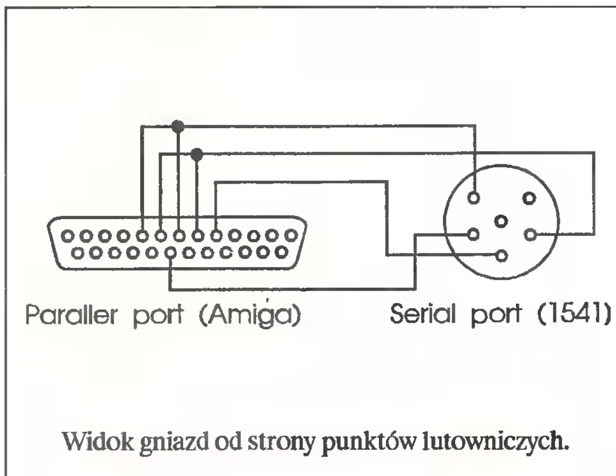
oznacza, że należy podać początkowy adres zbioru i jego długość.

Teraz już wystarczy zapisać zbiór na dysk i gotowe. Program skopiowany!

Sambor Kuźma

Amiga + C-64

Po ukazaniu się artykułu o symulatorze Commodore 64 na Amigę do redakcji napłynęło bardzo wiele listów. Większość z nich dotyczyła problemów z przenoszeniem programów z dysków 5.25" (C-64) na dyski 3.5". Najlepszym rozwiązaniem tego problemu jest zrobienie odpowiedniego kabla połączeniowego do stacji 1541 (standardowa stacja do C-64). Poniżej przedstawiam schemat takiego kabelka.



Widok gniazd od strony punktów lutowniczych.

Oprócz pokazanych połączeń należy również zewrzeć masy urządzeń, np. przez przylutowanie dodatkowej nitki przewodu do obudowy wtyczek.

Po wczytaniu symulatora należy wejść do opcji konfiguracji systemu (Ctrl+Help) i ustawić:

urządzenie 8 jako: C-64 Serial Cable 8
urządzenie 9 jako: DFO: Amiga Standard

Po wykonaniu tej operacji odwołanie się do urządzenia nr 8 będzie uruchamiało podłączoną stację 1541, a do urządzenia nr 9 wewnętrzną stację Amigi

Jak teraz kopiować programy? Robimy to w taki sam sposób, w jaki kopiowalibyśmy na C-64.

LOAD „nazwa",8 - ładuje program z dysku 5.25„
SAVE „nazwa",9 - nagrywa program na dysk 3.5"

Należy zwrócić uwagę na programy dłuższe - leżące powyżej adresu $A000. Ich skopiowanie jest niemożliwe bez odłączenia ROM Basic'a. Jeśli jednak będziemy używali jakiegokolwiek programu kopiującego kłopoty te nie wystąpią. Programy kopiujące mają jednak pewną wadę! Wiele z nich korzysta z tzw. fastów, czyli procedur przyspieszających pracę z dyskiem. Teoretycznie nie jest to żaden problem, tyle, że fasty nie działają z symulatorem!! Pojawia się więc kwestia znalezienia takiego programu kopiującego, który by nie był wyposażony w fast'a. Postanowiliśmy przyjść z pomocą Czytelnikom, i na jednym z kolejnych dysków PDP Amiga pojawi się odpowiedni kopiak. Wtedy przenoszenie programów nie sprawi już nikomu kłopotu.

Miłej zabawy ze starym, dobrym C-64 życzy:

Sambor Kuźma

Sculpt Animate 4D cz.3

Menu

Po uruchomieniu programu Sculpt 4D mamy do dyspozycji pięć menu, z których możemy wybierać opcje za pomocą myszki. Są to kolejno:

a) Project
b) Edit
c) Tools
d) Observer
e) World

Opis opcji dostępnych w menu zaczniemy od końca.

World

Za jego pomocą możemy zdefiniować przestrzeń, w której umieszczony będzie nasz obiekt. Na menu World składają się trzy sub-menu:

- *Sky*: możemy ustalić kolor nieba (None - czarne tło, Solid - niebo w jednym kolorze, Graduated - niebo cieniowane, wybieramy dwa kolory),
- *Ground*: ustawiamy kolory płaszczyzny ziemi (None - brak, Solid - ziemia w jednym kolorze, Checkered - ziemia kratkowana, szachownica),

Ilumination: ustawiamy kolor oświetlenia.

Observer

To menu dotyczy kreślenia gotowego rysunku i ustawiania parametrów dla niego:

- *Mode*: za pomocą tej opcji możemy ustalić jakość kreślonego rysunku (od Wire Frame - linia łącząca wierzchołki, do Photo - najlepszej jakości, uwzględniającej powierzchnie lustrzane i szklane), typ rysunku (Lo-res, Hi-res, Interlace, ilość bitplane'ów),
- *Location*: ustawia obserwatora w edytowanej przestrzeni na pozycji kursora,
- *Target*: w pozycji kursora ustawia punkt, na który patrzymy,
- *Lens*: wybieramy typ obiektywu, przez który patrzymy (Narmal - standardowy, Wideangle - szerokokątny, Telephoto - teleobiektyw, Special - ustawiamy własną ogniskową),
- *Image Size*: Pozwala ustalić wielkość otrzymanego rysunku (od Tiny -1/16 ekranu, do Jumbo - pełen overscan),
- *Tilt*: ustawiamy nachylenie obserwatora względem płaszczyzny ziemi,
- *Start* (klawisz F8): rozpoczyna proces kreślenia obrazu,
- *Display*: możliwość podglądania rysunku podczas tworzenia (Early), lub obejrzenia go po zakończeniu kreślenia (Late),
- *Abort* (klawisze CTRL i C): przerywa proces kreślenia,
- *Status* (klawisz F10): podaje informacje o obiekcie, wolnej pamięci, etc.

W następnym odcinku przedstawimy dalsze menu Sculpt'a 4D.

Marcin „Duddie" Dudar

Kącik początkującego kodera (cz. 3)

W dzisiejszym odcinku poznamy nowe istrukcje mikroprocesora MOTOROLA 68000. A więc zaczynamy...

ADD <adres efektywny>, Dx „Add Binary" - dodawanie binarne.

Dx,a<dres efektywny>

Instrukcja ta dodaje operand źródłowy do operandu przeznaczenia, wynik pozostawiając w operandzie przeznaczenia. Operacja może być wielkości bajtu, słowa lub długiego słowa.

Znaczniki: X - ustawiany, gdy wystąpi przeniesienie;
N - ustawiany, gdy w wyniku otrzymaliśmy liczbę ujemną, tzn. najstarszy bit wyniku jest ustawiony;
Z - ustawiany, gdy wynikiem jest zero;
V - ustawiany, gdy wystąpił nadmiar, tzn. wynik nie może być poprawnie zapisany na operandzie o podanej wielkości;
C - ustawiany, gdy wystąpi przeniesienie.

W przeciwnych wypadkach do wyżej opisanych znaczniki zostaną skasowane.

Przykład:

add.l d0,d7 - rejestr d0 zostanie dodany do rejestru d7. Wynik zostanie umieszczony w d7.

ADDA <adres efektywny>,Ax „Add Address" - dodaj adres.

Instrukcja ta dodaje operand źródłowy do rejestru adresowego i w nim też umieści wynik. Wynikiem tej operacji może być słowo lub długie słowo.

Znaczniki: pozostają bez zmian (jak w ADD).

Przykład:

adda.l (a0),a1 - adres znajdujący się w komórkach pamięci wskazanych przez rejestr a0 zostanie dodany do rejestru a1. Suma zostanie umieszczona w rejestrze a1.

adda.l a0,a1 - adres umieszczony w rejestrze a0 zostanie dodany do adresu umieszczonego w a1. Wynik tej operacji zostanie zapisany w rejestrze a1.

ADDI #liczba,<adres efektywny> „Add Immediate" - dodawanie natychmiastowe.

Instrukcja ta doda daną liczbę do operandu przeznaczenia i tam umieści wynik. Możemy wykonywać operacje na bajcie, słowie lub długim słowie.

Znaczniki: jak w ADD.

Przykład:

addi.b #$20,-(a0) - rejestr a0 zostanie zmniejszony o 1, następnie liczba $20 zostanie dodana do bajtu znajdującego się pod adresem wskazanym przez a0.

ADDQ #liczba,<adres efektywny> „Add Quick" - dodawanie szybkie.

Instrukcji tej używamy, gdy do operandu przeznaczenia chcemy dodać liczbę z zakresu od 1 do 8. ADDQ wykonywana jest znacznie szybciej niż instrukcja ADDI, dlatego koderzy (którym zawsze brakuje czasu) bardzo często ją stosują do zwiększania liczników pętli lub indeksów tablic sinusów.

Znaczniki: jak w ADD.

Przykład:

addq.w #2,$4000 - słowo znajdujące się pod adresam $4000 zostanie zwiększone o 2.

ADDX Dx,Dy „Add Extended" - dodawanie rozszerzone.

-(Ax),-(Ay)

Instrukcja ta dodaje operand źródłowy do operandu przeznaczenia wraz ze znacznikiem X. Wynik zostanie umieszczony w operandzie przeznaczenia. Rozmiarem tej operacji może być bajt, słowo lub długie słowo. Istnieją tylko dwa sposoby adresowania tej instrukcji: obydwa operandy mogą być rejestrami danych, albo można zastosować tryb predekrement.

Znaczniki: jak w ADD.

Przykład:

addx.w d0,d7 - młodsze słowo rejestru d0 zostanie dodane do rejestru d7. Do wyniku zostanie dodany bajt rozszerzenia X. Ostateczny wynik zostanie umieszczony w rejestrze d7.

AND <adres efektywny>,Dx „Logical AND" - 'i' logiczne

Dx,a<dres efektywny>

Za pomocą tej instrukcji możemy wykonać logiczny iloczyn dwóch liczb. Rozmiarem operacji może być bajt, słowo lub długie słowo. Iloczyn logiczny testuje odpowiednie pary bitów dwóch operandów i daje w wyniku 1 tylko wtedy, gdy oba testowane bity były ustawione. W przeciwnym wypadku wynikiem iloczynu logicznego jest 0.

0 i 0 = 0
0 i 1 = 0
1 i 0 = 0
1 i 1 = 1

Wartość iloczynu logicznego umieszczana jest w operandzie przeznaczenia.

Znaczniki: X - nie zmieniany;
N - ustawiany, gdy najstarszy bit wyniku jest ustawiony;
Z - ustawiany, gdy wynikiem operacji jest zero;
V, C - zawsze zerowane.

Przykład:

and.b d0,d1 - załóżmy, że w d0 mamy wartość $35, a w d1 $0F. Po wykonaniu tej operacji otrzymamy następujący wynik:

```
  %00110101 $35
i %00001111 $0F
= %00000101 $05
```

który zostanie umieszczony w rejestrze przeznaczenia d1.

ANDI #liczba,<adres efektywny> „AND immediate" - natychmiastowy iloczyn logiczny.

Instrukcja ANDI wykonuje iloczyn logiczny danej liczby z operandem przeznaczenia, gdzie też pozostawia wynik. Rozmiarem operacji może być bajt, słowo lub długie słowo.

Znaczniki: jak w AND.

Przykład:

Za pomocą tej instrukcji możemy np. sprawdzić czy trzy dowolne bity operandu są ustawione.

```
andi.b #%00101001,d0
beq    skasowane
```

Pierwsza instrukcja wykonuje logiczne And z bajtem rejestru d0. Jeżeli bity 0,3,5 będą skasowane to zostanie wykonany skok do etykiety „skasowane", natomiast jeżeli przynajmniej jeden z bitów będzie ustawiony, to wykonana będzie następna instrukcja.

ROL #liczba,Dx „Rotate left" - obrót w lewo.
Dx,Dy <adres efektywny>

Instrukcja ta wykonuje obrót operandu przeznaczenia o określoną liczbę bitów. Operand przeznaczenia może mieć wielkość bajtu, słowa lub długiego słowa. Co to jest obrót operandu? Załóżmy, że wykonujemy operację na bajcie. Wiemy, że liczbę zapisaną w bajcie stanowi osiem bitów ponumerowanych kolejno od b0 do b7. Po wykonaniu obrotu operandu w lewo o 1 bit b0 wejdzie na miejsce bitu b1, b1 na miejsce b2, b2 na miejsce b3,..., b7 na b0 - oraz dodatkowo do znacznika C. Aby lepiej to zrozumieć przyjrzyjmy się rysunkowi:

```
C . <---|b7|b6|b5|b4|b3|b2|b1|b0|<---
    >------------>--------->
```

Za pomocą tej instrukcji możemy nie tylko obracać bity o jeden, ale również o dowlną liczbę bitów z zakresu od 1 do 32 - związane są z tym trzy formaty tej instrukcji:

* natychmiastowy - obracamy w lewo o liczbę bitów z zakresu od 0 do 8,
* rejestru danych - liczbę bitów, o którą mamy obrócić operand przeznaczenia określa zawartość rejestru danych operandu źródłowego. W tym przypadku możemy obrócić operand o liczbę mieszczącą się w zakresie od 1 do 32.
* adresu efektywnego - obracamy w lewo o 1 bit.

Znaczniki: X - nie zmieniany;
N - ustawiany, gdy najbardziej znaczący bit jest ustawiony;
Z - ustawiany, gdy wynikiem jest zero;
V - zawsze zerowany;
C - ustawiany zgodnie z ostatnio wysuniętym bitem.

Przykład:

rol.l #2,d0 - obracamy rejestr w lewo o dwa bity,
rol.w d1,d7 - słowo rejestru d7 zostanie obrócone w lewo o ilość bitów zawartą w adresie d7,
rol.b $4000 - zawartość komórki $4000 zostanie obrócona w lewo o 1 bit.

Zapoznajmy się teraz z przykładowymym programem. Służy on do zamiany liczby, umieszczonej uprzednio w rejestrze d0 , na ciąg znaków ASCII, który zostanie wpisany pod adres za etykietą „Bufor". Procedura ta może być wykorzystana w programach przeliczajacych liczby z różnych układów numerycznych na heksadecymalne.

```
Start:
 lea       HexTab(pc),a0      ;adres tablicy znaków do a0
 lea       Bufor(pc),a1       ;adres bufora do a1
 move.b    #8,d7              ;wartość 6 umieszczamy
                              ;w rejestrze d7 jako licznik
                              ;pętii. Dlatego 8, że liczby
                              ;hex składają się z ośmiu
                              ;cyfr.
Pętla: rol.l  #4,d0           ;obrót rejestru d0 o 4 bity
                              ;cztery ostatnie bity staną
                              ;się pierwszymi
 clr.w     d1                 ;czyści młodsze słowo
                              ;rej. d1
 move.b    d0,d1              ;przesyła bajt z d0 do d1
 and.b     #%00001111,d1      ;„maskujemy" 4 młodsze
                              ;bity wyodrębniając naszą
                              ;cyfrę, która będzie
                              ;indeksem nast. instrukcji
 move.b    (a0,d1.w),(a1)+    ;w zależności od zawartości
                              ;d1 przepisujemy znak pod
                              ;adres wskazany przez a1.
                              ;Następnie adres zostaje
                              ;powiększony o 1.
 subq.b    #1,d7              ;odejmujemy 1 od d7
 bne       Pętla              ;jeśli zawartość d7 różna
                              ;od zera to skok do „Pętla"
 rts                          ;w przeciwnym wypadku
                              ;powrót
HexTab: dc.b „0123456769ABCDEF"
Bufor:  blk.b 8,0
```

Pamiętajmy, że liczbę z zakresu od $0 do $F można zapisać jako %0000 do %1111. Przedstawiona procedura wykorzystuje ten fakt wyodrębniając kolejne osiem czwórek bitów (począwszy od najstarszej), co w rezultacie daje nam całe 32 bity.

W pierwszej części programu następuje inicjalizacja - potrzebne wartości zostają umieszczone w rejestrach. W głównej pętli wyodrębniamy czwórkę bitów, następnie zamieniamy ją instrukcją '„move.b" na odpowiadający jej znak, który z kolei umieszczony zostaje w buforze. Jeżeli w d7 pojawi się zero, znaczy to, że zamienione zostały wszystkie cyfry heksadecymalne - następuje wtedy powrót. Jeżeli w d7 wartość jest różna od zera, to procedura „skacze" do etykiety „Pętla" i przekształca następną cyfrę.

Teraz liczbę, którą chcemy przekształcić możemy umieścić w rejestrze d0 i uruchomić procedurę. Po jej wykonaniu sprawdźmy zawartość komórek pamięci począwszy od etykiety „Bufor" - znajdziemy tam naszą liczbę zapisaną jako ciąg znaków.

Krzysztof „K.K" Kobus

KICKSTART 2.0 - TEST

Konstruktorzy Amigi 3000 wpadli na pomysł wyposażenia jej w nową wersję systemu operacyjnego (Kickstart'a). Nie to jednak jednak Kickstart 1.4 (jak się wszyscy spodziewali) ale nowy Kickstart 2.0.

W nowy system operacyjny można wyposażyć każdą Amigę. Jest on dostępny zarówno na kościach ROM jak i na dyskach. Zajmuje on 512 KB i w przypadku wersji dyskowej potrzebujemy 512 KB pamięci FAST (adres $c00000), w której to zostanie on umieszczony.

Po uruchomieniu Kickstart'a 2.0 zamiast starej, ohydnej „rączki" ukazuje nam się kolorowy znaczek Amigi oraz stacja dysków. Na ekranie animowana jest także dyskietka wkładana do tej stacji (to chyba wypełnia z połowę ROM ?).

Po włożeniu dyskietki do stacji na ekranie ukazuje się nam zupełnie nowe okno CLI z wbudowaną nakładką typu Shell.

Możemy teraz za pomocą kursorów „góra" i „dół" wywołać poprzednio wpisane komendy. Ciekawostką jest to, że każde okno ma swoją własną pamięć tekstów i jeżeli je skumulujemy to teksty są automatycznie formatowane. Każdy ekran ma teraz tylko jeden gadżet zamiast dwóch, za pomocą którego możemy ekran chować za pozostałe jak i przyciągać do przodu. Okno natomiast ma standardowo trzy gadżety, jeden do skalowania, drugi do zmiany priorytetu widoczności oraz trzeci do zmiany wymiaru (okno zamknięte - otwarte). Zamiast dawnego oznaczenia „siateczką" okna nieaktywnego mamy okna wypełnione kolorem nr 3 co poprawia efekt wizualny ale zwalnia pracę komputera (przy przełączaniu okien).

Bardzo dobrym posunięciem jest zainstalowanie „normalnych" fontów (Siesta) zamiast Topaz'ów, dzięki czemu teksty są bardziej czytelne. Nareszcie zainstalowano w ROM'ie Ram-Handler dzięki czemu nie musi być on stale doczytywany z dysku gdy chcemy używać Ramdysku.

Nowy Kickstart jest o wiele szybszy już na zwykłej Amidze. Zostały poprawione biblioteki graphics i intuition dzięki czemu wyświetlanie tekstów, rysowanie obiektów etc. jest błyskawiczne.

Całkiem nowa jest obsługa dysku. Zbiory ładowane są szybciej a prędkość wczytywania katalogów powiększona została około dziesięciokrotnie. Głowica mniej „lata" po dysku jednak loader jest nieco głośniejszy.

Nowy Workbench 2.0 to bardziej ciekawostka niż narzędzie pracy. Ikonki są wyświetlane w ramkach, czyniących Workbench trójwymiarowym i oświetlonym. Nowe komendy z katalogu C (Command) zostały napisane na nowo za pomocą amigowskiego C więc są o wiele krótsze i szybsze. Preferencje zostały rozbite na wiele krótkich programików i rozszerzone o wiele nowych funkcji. Workbench został dodatkowo wyposażony w wiele nowych bibliotek, między innymi w biblioteki AREXX'a, zostało także wzbogacone menu Workbencha, możemy na przykład wyświetlić zbiory jako ikonki do których nie mamy zbiorów info na dysku.

Zmiany uczynione w Kickstarcie 2.0 powodują, że część programów nie chce z nim pracować, m.in. te, które odwołują się bezpośrednio do systemu.

Dobrze napisane programy, korzystające w odpowiedni sposób z intution library oraz z innych bibliotek będą pracowały poprawnie z nową wersją Kickstarta. Z programów muzycznych dostępnych u nas najlepiej sprawował się ProTracker V1 1b napisany przez grupę Freelancers. Wszystkie inne Trackery zawiesiły się od razu. Program Oktalyzer zawiesza się oczekując na wyłączenie napędu stacji (jest to zapewne wina innej budowy struktury IORequest dla dysku). Będąc przy programach muzycznych należy omówić jeszcze playery (grajki) do muzyczek. ReqPlayer 3.0 chodzi bez zarzutu natomiast inny, bardzo użyteczny grajek - IntuiTtacker 1.23 czasami się zawiesza. Program do obróbki sampli - AudioMaster nie wykazywał żadnych odchyleń chociaż nie przetestowałem opcji samplowania dźwięku.

Programy kopiujące (X-Copy, White Lightning) działają bez zarzutu natomiast program DiskMaster 1.3 wogóle się nie uruchomił, działają za to wersje 1.4 i 3.0 tego programu.

Z Kickstarem 2.0 współpracują bez zarzutu takie programy jak: PowerPacker 3.0, MasterSeka 1.80, D-Mon Pro V1 7, i wiele innych (ten tekst pisany jest na edytorze CED uruchomionym właśnie pod Kickstartem 2.0).

Kickstart 2.0 można uznać za bardzo dobry system operacyjny. W zestawieniu z wielką szybkością Amigi 3000 jest rewelacją ale także dla użytkowników zwykłych A500 jest czymś na co warto wydać parę marek.

Marcin „Duddie" Dudar

PRZEGLĄD GIER

Cadaver

Z DZIENNIKA ANZELMA

Dzień trzeci

To już drugi dzień jak tu wszedłem. Dwa dni bez słońca. Długo wiosłowałem przez czarne, mokre bagno aż dotarłem do podnóża skały na której stoją ruiny. Zajdowało się tam tajemne przejście, pochodzące jeszcze z czasów panowania Wulf'a, do starych kopalni.

Wszystkie przejścia tworzyły wielki labirynt i wszystkie miejsca, które wcześniej brałem za charakterystyczne - kamienne narożniki, metalowe obręcze na ścianach, ścieżki porośnięte wielkimi grzybami - są dla mnie zupełnie obce. Wszędzie spotykam znaki śmierci: kości i stare ubrania ludzi, których prawdopodobnie znałem wiele lat temu kiedy Carolus został zamordowany. Wszystko już zdążyło zatrzeć się w pamięci. Czas jest nieubłagany...

Wiele rzeczy zdążyło się zmienić. Pamiętam kiedy kopalnia była pełna niawolników, kiedy praca wrzała, kiedy wszędzie było słychać dźwięki kilofów i przekleństwa. Teraz jest tu bardzo cicho, za bardzo...

Dzień piąty

Kolejne nowe odkrycie. Wczoraj wydawało mi się, że słyszałem dźwięki jakby wielkiego, rozpalonego pieca hutniczego, lub gigantycznego zwierzęcia które wyje by je uwolniono. Nie byłem jednak zupełnie pewny. Dzisiaj upewniłem się, że te dźwięki były prawdziwe. Siedziałem nad cichym jeziorkiem jedząc posiłek kiedy ryk znowu przetoczył się przez korytarze. Słyszałem go całkiem wyraźnie. Ryk przyniósł ze sobą ledwo wyczuwalny powiew...ciepły i kwaśny.

To tylko niewielkie wyjątki z dziennika Anzelma, wielkiego wojownika, który zakradł się do zamku Wulf'a po to aby... No właśnie, po co ? Tego nikt nie wie. Nie wiadomo również jak się zakończyła jego przygoda, gdyż w znalezionym dzienniku brakowało wielu kartek. Ostatni zapisek dotyczył skarbca po brzegi wypełnionego złotem i drogimi kamieniami. Jaskinie czekają na Ciebie !

Grafika:8
Muzyka: 7
Animacja: 6
Pomysł: 3
Przyjemność grania: 7
Ogólnie: 6.2
Wydawca: The Bitmap Brothers (Image Works)
Komputer: Amiga 500 (1MB); 2 dyski

Elvira, Mistress of the Dark

SUPER, HYPER, MEGA-BLAST !!! Tak zazwyczaj określa się gry które wybijają się ponad przeciętną. Elvira wybija się jeszcze wyżej.

Jest to jedna z niewielu gier, które potrafią zachwycić swoją zawartością do tego stopnia, że człowiek traci kontakt z rzeczywistością. Graficznie gra jest dopracowana do perfekcji. Tajemnicza i pełna groźby muzyka dopełnia scenerii. Programiści spisali się na medal: animowane przez nich obiekty są naprawdę duże i poruszają się na ekranie płynnie!

Jako młody chłopak trafiłeś do dziwnego zamku w którym wszyscy są poprzebierani w średniowieczne stroje, walczą mieczami i używają czarów.Zanim jednak na dobre zdążyłeś przekroczyć bramę zostałeś złapany i wtrącony do więzienia. Po kilku strasznych godzinach, kiedy już zdążyłeś stracić wszelką nadzieję, drzwi twojej celi nagle otworzyły się i zobaczyłeś w nich piękną kobietę, która wybawiła cię z opresji Zanim jednak zdążyłeś jej podziękować, okazało się, że to ona żąda od ciebie pomocy! Musisz jej pomóc odzyskać zamek, którym zawładnęli dziwnie wyglądający obcy pod władzą Emeldy — konkurentki twojej pracodawczyni. Musisz więc nauczyć się posługiwać bronią i czarami, które zresztą czasami są lepsze od stali!

Każdej lokacji w grze towarzyszy osobny obrazek. Daje to wspaniałe złudzenie rzeczywistego porusznia się po zamku i okolicach. Do każdej z sytuacji dopasowana jest odpowiednia muzyka. Inna jest w fosie (efekty płynącej wody) a inna np. w kryptach (tajemnicza i groźna).

Stan zdrowia gracza opisuje sześć wskaźników, które zmieniają się w czasie gry. STR (strenght) to twoja siła, RES (resistance) to odporność na ciosy (zwiększy się np. jeżeli będziesz używał tarczy), DEX (dexterity) to zręczność, SKI (skill) to umiejętność w posługiwaniu się mieczem, LIF (life) to twoje siły życiowe i na koniec EXP (experience) czyli doświadczenie.

Nie jest to gra dla miłośników „rąbania co popadnie".Czasami trzeba sporo pomyśleć nad rozwiązaniem niektórych problemów! Jednak wszystkim tym, którzy chcą na długie godziny pogrążyć się w świecie Elviry gorąco polecam tę grę.

Grafika: 10 (!)
Muzyka/FX: 8
Animacja: 9
Pomysł: 10 (!)
Przyjemność grania: 10 (!)
Ogólnie: 9.4 (super plus!)
Wydawca: Accolade (Horror Soft)
Komputer: Amiga 500 (1MB): 5 dysków

Jeśli któremuś z czytelników udało się ukończyć Elvire to jest proszony o niezwłoczny kontakt z redakcją (dopisek-Elvira). Uratuje w ten sposób niżej podpisanego od zupełnej utraty kontaktu z rzeczywistością.

Panza Kick-Boxing

Jeszcze jedna gra typu „trzaśnij go w mordę". Tym razem jednak nadzwyczaj sprawnie zaprogramowana, z piękną animacją i ładną grafiką. Gracz do wyboru ma aż 55 (pięćdziesiąt pięć) ciosów !!!

Wszystkich oczywiście nie da rady obsłużyć za pomocą jednego joystick'a. Dlatego można sobie wybrać ich tylko 13. Komputer wystawia do walki ośmiu przeciwników. Aby jednak można było walczyć z samym wielkim mistrzem trzeba się dobrze namęczyć z poprzednikami. „Przechodząc" ich po kolei gracz zbiera różnego rodzaju trofea (np. puchary). Stan gry komputer sam zapisuje na dysk. Aby poprawić swoje osiągnięcia autorzy dodali specjalną opcję treningu. Skakanie przez skakankę, podnoszenie ciężarów i kopanie w słupki podnosi refleks, siłę i odporność na ciosy.

W sumie gra jest najzabawniejsza gdy gra się z kolegą. Niżej podpisany wraz z całą redakcją (a w zasadzie przeciw niej) już od miesiąca walczy o zdobycie Wielkiego Pasa.

Grafika:	**8**
Muzyka/FX:	**5**
Animacja:	**10 (!)**
Pomysł:	**4**
Przyjemność grania:	**10 (!)**
Ogólnie:	**7.4**
Programowanie:	**Nicolas Massonnat**
Grafika:	**Marco De Flores**
Muzyka:	**Michel Winogradoff**
Komputer:	**Amiga 500 (1MB): 2 dyski**

Continental Circus

Jeszcze jedna konwersja z gier Coin-Op na Amige. Tym razem są to wyścigi samochodów formuły pierwszej. Pomysł stary jak świat, wykonanie bardzo niestarannie. Gra niczym się nie wyróżnia z pośród setek jej podobnych. W sumie szkoda miejsca w naszym piśmie na jej opis.

Grafika:	**3**
Muzyka/FX:	**3**
Animacja:	**2**
Pomysł:	**1**
Przyjemność grania:	**2**
Ogólnie:	**2.4 (dno)**
Programowanie:	**Pete Hickinson**
	Bill Caunt
Grafika:	**Mark Edwars**
Wydawca:	**Sales Curve (Taito)**
Komputer:	**Amiga 500: 1 dysk**

Recenzujący: Sambor Kuźma

*** Virus Expert v1.6 ***

Virus Expert 1.6 jest jednym z programów zamieszczonych na naszym drugim dysku PDP. Jest to jeden z lepszych tzw. "Virus killer'ów" czyli programów przeznaczonych do łapania i unieszkodliwiania wszelkiej maści wirusów. W wielu listach pojawiały się prośby o podanie chociaż skróconej instrukcji do programu. Zapraszam więc wszystkich "morderców" do lektury.

Najpierw jednak słów kilka o autorach. Starszą wersję (v1.4) programował Paul van der Valk. Do kodu dokopał się jednak jeden z lepszych (jeśli nie najlepszych) polskich koderów - Maciek Marzec (Mac z Katharsis). Wprowadzi kilka istotnych poprawek i tak powstał Virus Expert v1.6. Obecnie Maciek pracuje nad wersją 2.0, która będzie już programem komercyjnym.

Po wczytaniu programu Virus Expert automatycznie doczytuje tzw. library czyli bibliotekę wirusów. Aktualnie w mojej wersji biblioteka ta zawiera 101 bootblock'ów. Nie oznacza to, że jest to 101 wirusów. Niektóre z bootblock'ów są bardzo dobrymi programami do ich wykrywania i niszczenia. Kiedy biblioteka zostanie doczytana Virus Expert oczekuje na rozkazy użytkownika (komunikat: By your command). Teraz już jesteśmy gotowi do zabijania "niewiernych", a trzeba przyznać, że robi się to z wielką łatwością.

Program potrafi wykrywać i unieszkodliwia dwa typy wirusów. Pierwszym z nich jest najczęściej spotykany tzw. bootblock virus. Wirusy tego typu znajdują się na tzw. bootblock'u i są uruchamiane przy każdym starcie komputera z zarażonej dyskietki. Aby wykryć takiego wirusa wystarczy po uruchomieniu Virus Experta włożyć zarażony dysk do stacji. Po chwili na ekranie ukaże się komunikat o znalezionym i wczytanym bootblock'u. W okienku "Label" program wydrukuje nazwę ewentualnego wirusa. Jeśli nie jesteśmy pewni czy wczytany bootblock jest wirusem warto użyć opcji "ANALYSE", która przeanalizuje dany kod i zadecyduje czy jest to wirus. Oczywiście nie należy bezgranicznie ufać jej decyzjom. Czasami zupełnie niewinne programy są określane jako wirusy natomiast wirusy jako niewinne programy. Mimo tego jeśli komunikat brzmi:

It is possibly a virus

lub

It is very likely a virus Might be dangerous

nie należy się długo zastanawiać i zabić żyjątko.

Do usuwania wirusów bootblock'owych służy opcja "Install". Po jej włączeniu (należy kliknąć na "ON" - komunikat "Install enabled") do wyboru mamy trzy opcje: STANDARD, NO BOOT i BUFFER. Kliknięcie na pierwszą z nich spowoduje zapisanie na dysku standardowego dos'owego bootblock'u. Druga uczyni dysk "niebootowalnym", a trzecia spowoduje zapisanie bootblock'u znajdującego się w tzw. buforze. Aby to "coś tam" załadować trzeba kliknąć na opcję "LOAD BUFFER", następnie wybrać dany bootblock. Jak się domyślacie może to być zarówno wirus jak i program do zabijania wirusów. Można więc za pomocą Virus Experta również zarażać dyski!

Ze wszystkich programów sprawdzających obecność czegoś niepożądanego w pamięci komputera polecam "KTS Checker v3.2". Sprawdza on nie tylko pamięć ale również system ekranu w jakim uruchomi się komputer (NTSC lub PAL).

Virus Expert może również szukać i usuwać z dysków drugi rodzaj wirusów - tzw. wirusy "linkujące się". Do tego celu służy opcja "CHECK LINK". Wykrywa ona i usuwa trzy najbardziej znane wirusy: IRQ, BGS9 i BUTONIC. Jak jej używać? Po włożeniu dyskietki, którą chcemy sprawdzić, należy kliknąć na CHECK LINK. Na ekranie otworzy się jeszcze jedno okienko w którym komputer będzie drukować efekty poszukiwań. Jeśli wirus zostanie zdemaskowany maszyna poprosi o potwierdzenie chęci usunięcia paskudztwa (należy wcisnąć "Y" lub "N"). Niestety Maciek nie ustrzegł się tu pewnego błędu. Przy usuwaniu wirusa o nazwie BUTONIC komputer nie zmienia zawartości Startup-Sequence i w efekcie po zabootowaniu odkażonego dysku pojawia się komunikat o błędzie: Unknown command.

"Check link" jest również wyposażone w opcję "AUTO", której włączenie powoduje automatyczne sprawdzanie obecności wirusów linkujących się zaraz po włożeniu dysku do stacji.

Dwie opcje "VIEW ASCII" i "VIEW HEX" służą do przeglądania zawartości bootblock'u. Pierwsza powoduje wyświetlenie go jako znaków ASCII (czyli tekstów), druga do liter dodaje ich odpowiedniki w kodzie szesnastkowym.

Jeżli znajdziecie na dysku nowego wirusa możecie również zapisać go w waszej bibliotece. Do tego celu służy opcja RECORD BB. Ma ona możliwość zapamiętywania bootblock'ów jako samych nagłówków (Remember header only) lub jako całego kodu (Remember entire bootblock). Przełączania systemu zapisu dokonuje się przez kliknięcie na "FULL".

Czym różnią się te dwa systemy? Jeśli zapiszemy bootblock jako sam nagłówek oszczędzimy miejsce w bibliotece lecz potem nie będziemy mogli zgrać samego kodu na dysk. Zapis w postaci pełnego BB zabiera więcej miejsca lecz pozwala używać w przyszłości zapamiętany bootblock. Aby zgrać całą bibliotekę na dysk wystarczy kliknąć na okienko SAVE LIBRARY. Program jest również wyposażony w możliwość usuwania BB z biblioteki - trzeba nacisnąć klawisze SHIFT i D.

Aby biblioteka zajmowała mniej miejsca na dysku można ją skompresować za pomocą Power Packer'a. Virus Expert rozpoznaje automatycznie czy jest ona sprężona czy nie i w zależności od jej stanu podejmuje odpowiednie działanie

Część opcji zainstalowanych w programie jest dostępna tylko z klawiatury. Aby uzyskać pełny spis tych funkcji wystarczy nacisnąć klawisz K. Program ma również funkcję HELP. Po jej włączeniu klikanie na odpowiednie okienka zamiast ich wykonywania będzie powodowało wyświetlenie ich opisu.

Mam nadzieję, że teraz blady strach padnie na wszystkie szczepy wirusów, a Wam będzie się je przyjemniej mordowało!

Sambor Kuźma

P.S. W następnych wydaniach Public Domain zamieścimy kilka innych programów do usuwania lub niszczenia wirusów.

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C-64

Styczeń

Strona A

- Mega demo grupy „VISION" - MIST2

Strona B

- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

Luty

Strona A

- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+ (DEMO) [02]
- DREPTACZ BASIC

Strona B

- SWISS CHEESE/CFA
- DISK FAST LOADER

Marzec

Strona A

- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELOKATOR 64
- KOREKTOR 64
- FLASH

Strona B

- HOT SHOT nr. 9 (zachodni magazyn fanów)
- BAD NEWS nr. 2 - j.w.
- DEMO - rekord - 290 sprite'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER - adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

Kwiecień

Strona A

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku.

Strona B

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika.

Maj

Strona A

- CRUEL SOLIDERS - demo
- DESTINATION'91 - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

Strona B

- MEGA DEMO „INFOSYSTEM 91"

PUBLIC DOMAIN PACK AMIGA

Styczeń

- Programy kompresorów danych
- Grafiki Borysa Vallejo
- Prezentacja najlepszych muzyczek
- INTUITRACKER

Luty

- Request player;Multi ripper
- 3-rd day; Phantasmagoria - demo
- Master Seka; Virus Ekspert v1.6
- AMOS - programy; Moduły: Killing game show, Upon Me, Let's swing it.

Marzec

- Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)
- Najlepsze muzyczki: NOW WAT? - DR. AWESOME
- AMOS - procedury
- DEMO grupy REBELS „TOTAL TRIPLE TROUBLE"

Kwiecień

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  - TECHNOSTYLE 2
  - GALAXY 2
- GRAFIKA - prezentujemy rysunki
  - RICK PARKS

Maj

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  - MIAMI VOICE
  - ANTI ATARI SONG

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr:5.09011-400522.7-136-11-111.0 następujących kwot: 20.000zł za pojedyńczy zestaw dla C-64, 25.000zł za zestaw dla AMIGI. Kwoty te obejmują koszt dyskietki, koszty kopiowania, opakowania i przesyłki pocztowej. Blankiety wpłat powinny być **CZYTELNIE** wypełnione i zawierać: **Imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64"**(jeśli zamawiamy zestaw dla C-64) lub **„PDP-A** "(zestaw dla Amigi). W prenumeracie zestawy kosztują: PDP-64 - 18.000zł (12 numerów 216tys zł), PDP-A - 22.000zł (12 numerów 264tys zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Prenumerata może obejmować miesiące od początku roku - tzn. zamawiając całoroczną prenumeratę np. w lipcu, w pierwszej przesyłce otrzymacie wszystkie poprzednie zestawy (tj. od stycznia, do lipca).

ZAMÓW NIE ZWLEKAJ!

